Machine Learning Prediction of Power Demand for Electrical Vehicle Charging Stations in Norway

Thesis submitted in partial fulfillment of the requirements for the degree of
Master in Informatics: Programming and System Architecture
The Faculty of Mathematics and Natural Sciences
University of Oslo.

Khalil Abuawad

Department of Informatics, University of Oslo

Supervisors:
Mohsen Vatani, Institute for Energy Technology
Jonathan Fagerström, Institute for Energy Technology
Frank Eliassen, University of Oslo

November 2019

Abstract

With the growing popularity of electrical vehicles (EVs), many problems concerning the charging infrastructure have appeared. Developing strategies to manage this problem efficiently is essential for the charging infrastructure. One problem is the queuing times at EV charging stations; another is power-demand management. This thesis provides a broad view on how power-demand prediction with the help of machine learning (ML) can help solve these issues and more. The Institute of Energy Technique (IFE) chose the primary goal of this thesis to be a comparison of the ability of various methods to predict the power flows for the week ahead, for several important locations in Norway. The comparison includes six different models: a classical model called autoregression (AR); three ML models known as regression tree (RT), support vector regression (SVR) and K-nearest neighbour (KNN); and two deep learning (DL) models called recurrent neural network (RNN) and long short term memory (LSTM). Three main tests were performed for this thesis. These were as follows: 1) whether classical methods are accurate for EV-charging station power-demand prediction; 2) the difference between predicting the power demand for high activity versus low activity charging station routes; and 3) the effects of weekday binary, peak-day binary, temperature and X previous observations (X = 24 and 168) as parameters on the models' ability to predict. The thesis includes a literature review for real-time learning that may benefit future researchers.

Acknowledgements

I would like to thank all of my supervisors for giving me this opportunity to be a part of a project that is new and relevant in the world today. To experience the development of such an interesting project firsthand has been a great experience, and I will forever be grateful to have been a part of it. I would like specifically to thank my two supervisors at the Institute for Energy Technology, Mohsen Vatani and Jonathan Fagerström. Their support based on experience and their intellectual guidance have made this thesis possible. Thank you for sharing your knowledge and offering advice, and for each helpful discussion we had.

I would also like to thank my family and friends for helping me whenever I needed it.

Khalil Abuawad,
November 2019

Contents

List of Figures

Chapter 1

Introduction

Most of the world's greenhouse gas (GHG) emissions are the result of electricity generation, transportation and other forms of energy production and use [12]. This has led to a global increase in temperature and subsequently climate problems. In addition, the world's energy demand has increased, mainly due to continuous population growth, thus increasing the severity of the problem. To address this situation, global leaders have realized the need to save energy and to replace fossil electricity production with renewable energy.

Through insight that the transportation sector is a large contributor to GHG emissions [12] and the desire to move away from fossil fuels, electrical vehicles (EVs) and other low-emission vehicles were introduced. EV sales have increased every year and continue to grow, particularly in Norway. In 2015, Norway had a market share of 18%, whereas other countries were below 1%[10]. This is due to the Norwegian government creating a friendly environment for EV owners, with considerable incentives.

The growth in EV units has helped Norway to reduce its GHG emissions but has awoken scalability concerns regarding power demand for EVs. There are two main concerns that need to be addressed. The first is providing the required power at all charging stations at all times, and the second is reducing queuing times at the charging stations.

1.1 Motivation

In the near future there will be a dominance in EVs. EV owners will not tolerate long waiting times at charging stations and therefore queuing times need to be short. Politicians have set an ambitious goal for 2025, namely that all cars sold should be zero-emission. Norway set a world record for the number of EV cars sold in one month. According to the Director of the Road Traffic Advisory Council, 45.3% of all passenger cars sold September 2018 were EVs [36].

With increasing numbers of EV consumers, scalability and power-demand concerns arise. The choice of solutions to address these concerns is important for the future of the charging infrastructure. One study [48], that by using power-demand prediction, it was possible to reduce peak power demand, resulting in decreased daily power usage; in turn, this led to environmental and economic benefits. By accurately predicting the power demand, one can address the power demand more appropriately. Fast chargers (will be explained in the next chapter), have a vast impact on the power grid, which leads to low power availability. Therefore, accurate prediction can help to measure and control the storage capacity at charging stations so that grid companies can deliver the required power without overly expensive grid reinforcements. By informing the consumers of the charging stations status one would be utilizing charging stations even more, because the consumers would have the ability to plan their routes based on the charging stations status.

EV driving distance and recharging time of the battery are two main disadvantages of being an EV owner. Currently there are two approaches to compensate for these issues. They are larger batteries, that is, a higher energy density in batteries, and faster charging of EVs, that is, through "superchargers". It is common to see a third solution: owning two cars, one an EV and another fuel/hybrid-based.

There is a crossroad to increasing the size of EV batteries and charging them faster. These technological improvements do not necessarily go hand-in-hand. Two of the most promising high-energy cell technologies, namely cells that employ Si-enhanced or Li-metal negative electrodes, currently suffer significant degradation in cell life upon fast charging [45]. The question arises whether fast charging of a lithium ion (Li-ion) is superior to implementing a high-energy density cell that cannot be fast-charged.

Fast chargers are ideal for long distance traveling and there are indications that users prefer to use fast chargers throughout their travels. Studies show that having access to fast chargers increases the use of EVs [35]] and increases the traveling distance of EV owners by approximately 25% [16]. Although fast chargers are highly popular, they great impact on the power grid. Therefore, fast charging depends on the capacity of the local power grid to establish an optimal balance between fast and slow charging.

Many strategies exist for reducing the impact of fast charging on grids. An example is combining fast charging with energy storage that discharges at the time of peak demand. However, when installing a reserve energy storage beforehand, technicians lack information of how large the energy reserve should be. This is where power-demand prediction is useful.

1.2 Project Goals and Execution

The Institute for Energy Technology (IFE) is conducting a project called Integrated Transport and Energy Modelling (ITEM). The primary goal of ITEM is to determine policies and measures best suited to reach carbon neutrality in the transportation sector. The research question in the project is as follows: "What are the prerequisites and implications in terms of energy supply, power generation, local and regional grid distribution of fast charging and hydrogen production?"

To be able to predict power demand at a specific location is crucial for an efficient management of the charging infrastructure. Thus the primary goal of this thesis is to predict week-ahead power flows at two routes in Norway. The primary goal of this thesis is to predict week-ahead power flows at two routes in Norway. The secondary goal of this thesis was to compare autoregressive, machine learning and deep learning models for predicting the week-ahead power flows of two important routes in Norway. This was done by finding input parameters and features that the models can benefit from.

The thesis included three main tests. The first compared six models using univariate input. The six models included one classical model, an autoregressive (AR) model, which was included to gain insight on whether there is a need to use complex and sophisticated methods such as machine learning (ML) or deep learning (DL) for this particular problem, or if classical methods like AR are sufficient. The second test had the objective of gaining understanding about how different routes in Norway can influence the ML/DL models' forecasting abilities. The third test provide understanding on how all the different parameters found throughout the thesis effects these models. IFE also has a final goal for this thesis, which is to propose a solution for real-time learning and new types of data that can benefit future research. The research topic can be summarized as follows:

Multi-step Time-series Short-term Supervised Machine Learning Power Demand Prediction of EV Charging Stations in Norway.

1.3 Thesis overview

The remainder of the thesis is divided into the following chapters.

Background

The background chapter covers most topics that need to be addressed before the rest of the thesis content can be presented. The chapter starts by discussing charging stations and the topics relevant for the thesis. It then explains multi-step prediction and various strategies that can be used. The chapter goes on to explain time series in full, including converting a time-series forecasting problem into supervised learning; the sliding window technique; the difference between time-series analysis and time-series forecasting; and the difference between univariate and multivariate time series. The chapter ends with a short description of the prediction methods used in this thesis and their advantages and disadvantages.

Relevant literature

This chapter discusses literature about the prediction methods presented at the end of the previous chapter. The literature does not always include prior studies on this topic, which is new; in addition, some methods might not have been researched before within this research field. However, gaining understanding about how they have performed in other research fields proves interesting, nonetheless.

Data preprocessing and analysis

Descriptive information about the data used in this thesis, including data preprocessing and the handling of missing data and outliers. The chapter also analyzes the data, which includes inspecting the plug type amount and the hourly and daily power-demand patterns. Extra data included in the dataset are presented, as well as a short explanation of the charging station categorization.

Prediction models

Presents the models, tests, tools, and evaluation metrics, and their implementation which was used throughout the thesis.

Results and discussion

This chapter examines how the tests were conducted. Thereafter, the tests are presented one by one and the results are discussed.

Conclusions and recommendations

This chapter provides an analysis of the key results and a future perspective on the work.

Appendices

Chapter 2

Background

This chapter provides background information for Power Load Management, Time Series Forecasting and a literature review on previous work. This will help readers gain the extra information needed to understand the problem and topic of the thesis better.

There are currently 194 900 EV units in Norway, and this number will continue to grow. With the amount of EV units continuing to grow there are certain dilemmas that are being faced, as previously mentioned. EV charging stations usually consists of multiple outlets where each outlet usually give access to fast charger plugs and normal plugs, where only one of these types can be used at once. The charging station are typically divided into four different charging modes. These modes are given by The International Electrotechnical Commission under the standard IEC 62196.

2.1 Charging stations

2.1.1 Plug types

There are currently three different charging station levels. Level 1 charging stations use a 120 V AC plug and can be plugged into a standard outlet. Unlike other chargers, level1 chargers do not require the installation of any additional equipment. These chargers typically deliver two to five miles of range per hour of charging and are most often used at home. Level 2 chargers are used for both residential and commercial charging stations. They use a 240 V (for residential)or 208 V (for commercial) plug, and unlike level 1 chargers, they can't be plugged into a standard wall outlet. Instead, they are usually installed by a profssional electrician. Level 3, also known as DC Fast Chargers or CHAdeMO/CCS charging stations, can offer 60 to 100 miles of range for your electric car in just 20 minutes of charging. However, they are typically only used in commercial and industrial applications - they require highly specialized, high-powered equipment to install and maintain.

2.1.2 Charging station types

Insight of EVs being a very hot and new topic there has not been a lot of research concerning the types of EV charging stations. Being able to identify different feature in charging stations might help with producing more accurate prediction models for each charging station work with. In this paper [37], they categorized charging stations in three, specifically, Work, Home and Other. The category “Work” contained locations like workplaces, schools and universities. The “Home” category were residential locations. Lastly, “Other” which contained the rest of the locations. These were based on charging profiles represented in the paper, where “Work” locations were expected to have an early morning load profile mostly on weekdays. “Home” locations were expected to be a late afternoon or early evening charging load profile, and lastly, “Other” were expected to be a flat midday to late afternoon load profile. Although this paper somewhat categorize EV charging stations into three types, the only type that we have data of are of the type "Other". This type can probably be divided into smaller types, but this will not be covered in depth in this thesis.

Other than these three categorizations made in the paper presented, one can categorize the charging station by the speed of how fast one can charge, mostly known as fast chargers, or the regular charging stations with mostly type 2 plugs. A third way to categorize these charging stations is by popularity.

2.1.3 Power-demand

Being able to predict the power-demand at charging stations is very important for charging station owners economy, mainly due to peak power-demand levels where the cost for reinforcing electricity can be very expensive and therefore strategies for reducing these costs are immensely important for them. Being able to predict the power-demand and their peaks will help establish strategies for reduces them and their costs. Yet again, if these peaks are close to each other being able to resupply the reserves needs to be calculated. For some of the strategies regarding reducing peak loads, predicting the peak loads also means knowing when there is time to recharge a reserve of electricity. There are two types of approaches in dealing with peak loads, namely peak shaving and load shifting. Both will be explained further down.

2.1.4 Peak loads

Peak loads and grid usage fees go hand in hand, i.e. the way you tackle the peak loads has a huge impact on the grid usage expenses. Grid operators is not certainly happy about peak loads, mainly due to the fact that they must design the grid based on maximum amount of power that will be needed. Sudden load increases

can be reliably detected by monitoring the power consumption. These peak loads are used to calculate grid usage fees assessed to certain power consumers. Peak loads can also help identify times for reserves to be charged. This is important when it comes to peak shaving which will be described further down. By know when a reserve energy storage can be recharged, one can economically efficiently tackle this peak loads.

By taking the figure 2.1 as example, knowing the time when the power consumption is low, and take the amount of observations (time) multiplied the amount of available kW to recharge the reserves. In other words, by multiplying the time and kw available at the time between two peak loads, one can determine the amount that can be take from the grid to recharge the reserves.

2.1.5 Peak shaving

> *In the energy industry, peak shaving refers to leveling out peaks in electricity use by industrial and commercial power consumers. Power consumption peaks are important in terms of grid stability, but they also affect power procurement costs: In many countries, electricity prices for large-scale consumers are set with reference to their maximum peak-load. The reason is simple: the grid load and the necessary amount of power production need to be designed to accommodate these peak loads.* [32]

Peak shaving is when the power consumption is reduced quickly and for a small period of time, thus avoiding a spike in the consumption. Some main strategies for dealing with peak loads are load shifting and peak shaving. Each of this strategies can be done in multiple ways. Uddin et al. [44] did a review on peak load shaving strategies, namely the three major strategies for peak load shaving, demand side management (DSM), integration of energy storage system (ESS), and integration of electric vehicle (EV) to the grid. The study discovered these three to be the major peak shaving techniques, and identifies each strategy's unique challenges and further research and investigations that needs to be done.

2.1.6 Load shifting

> *In contrast, load shifting refers to a short-term reduction in electricity consumption followed by an increase in production at a later time when power prices or grid demand is lower. Dedicated generators or electricity storage facilities owned by the power consumer can be used to bridge high-price or high-load phases, but play less of a role if production will eventually catch up again.* [32]

The quote above explains how load shifting is the process of mitigating the effects of large energy load blocks during a period of time by advancing or delaying their

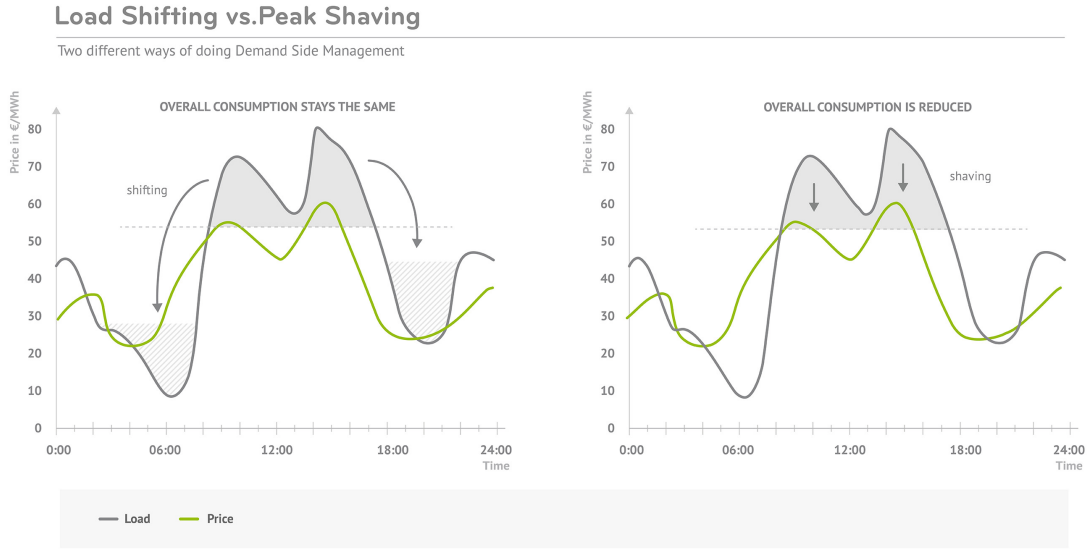


Figure 2.1 Load Shifting vs. Peak Shaving [32]

effects until the power supply system can readily accept additional load. Both load shifting and peak shaving can be used at the same time, Rozali et al. [38] aimed to achieve maximum peak shaving through Demand Response (DR) a load shifting strategy. In the study, Rozali proposed two load shifting strategies; (i) by reallocating the outsourced electricity to the time intervals with electricity surpluses occurring during off-peak hours, thus reducing the amount of outsourced electricity requirement during peak hours, (ii) electricity demand during peak hours can be shifted to the time intervals straddling the peak and off-peak hours, provided that the time interval where the demand is shifted to is preceded by the time interval with a large electricity storage.

2.2 Multi-step Prediction

The number of time steps ahead to be predicted is important. It is traditional to use different names for the problem depending on the number of time-steps to predict. One-Step Prediction is where the next time step $(t + n)$ is predicted, while Multi-Step prediction is where two or more future time steps are to be predicted.

2.2.1 Direct Multi-step Prediction Strategy

The direct method involves developing a separate model for each prediction time step. In the case of predicting the power-demand at a charging station for the next two days, we would develop a model for predicting the power-demand on day one

and a separate model for predicting the power-demand on day two. For example:

$$prediction(t+1) = model1(obs(t-1), obs(t-2), ..., obs(t-n)) \tag{2.1}$$

$$prediction(t+2) = model2(obs(t-1), obs(t-2), ..., obs(t-n)) \tag{2.2}$$

Having one model for each time step is an added computational and maintenance burden, especially as the number of time steps to be predicted increases beyond the trivial. Because separate models are used, it means that there is no opportunity to model the dependencies between the predictions, such as the prediction on day two being dependent on the prediction in day one, as is often the case in time series.

2.2.2 Recursive Multi-step Prediction

The recursive strategy involves using a one-step model multiple times where the prediction for the prior time step is used as an input for making a prediction on the following time step. In the case of predicting the power-demand at a charging station for the next two days, we would develop a one-step prediction model. This model would then be used to predict day one, then this prediction would be used as an observation input in order to predict day two. For example:

$$prediction(t+1) = model(obs(t-1), obs(t-2), ..., obs(t-n)) \tag{2.3}$$

$$prediction(t+2) = model(prediction(t+1), obs(t-1), ..., obs(t-n)) \tag{2.4}$$

Because predictions are used in place of observations, the recursive strategy allows prediction errors to accumulate such that performance can quickly degrade as the predication time horizon increases.

Examples of machine learning models using this Multi-Step strategy are recurrent neural networks[46][47]. An advantage of using recursive strategy is that only one model is required saving significant computational time, especially when a larger number of time series and prediction horizons are involved. The strategy also ensures that the fitted model m matches the assumed data generating process f as closely as possible. On the other hand, the recursive forecasts are not equal to the conditional mean, even when the model is exactly equivalent to the data generating process.

2.2.3 Direct-Recursive Hybrid Strategies (RECTIFY)

Based on this study they propose a new prediction strategy that seeks to combine the best properties of both the recursive and direct strategies. They call the strategy

"RECTIFY", where the rational behind this is to begin with biased recursive predictions and adjust them so they are unbiased and have smaller error. For example, a separate model can be constructed for each time step to be predicted, but each model may use the predictions made by models at prior time steps as input values. We can see how this might work for predicting the temperature for the next two days, where two models are used, but the output from the first model is used as an input for the second model. For example:

$$prediction(t+1) = model1(obs(t-1), obs(t-2), ..., obs(t-n)) \tag{2.5}$$

$$prediction(t+2) = model2(prediction(t+1), obs(t-1), ..., obs(t-n)) \tag{2.6}$$

Combining the recursive and direct strategies can help overcome the limitations of each. This is done by starting with a simple linear base model, and produce predictions from it using the recursive strategy. These are known to be biased, thereby correcting these prediction is needed. This is done by modelling the prediction errors using a direct strategy. The resulting prediction will be unbiased, provided the models used in the direct strategy are sufficiently flexible. The advantage of this two-stage process is that it links all the direct prediction models together with the same unifying base model, thus reducing the irregularities that can arise with independent models, and so reducing the prediction variance. The paper discovered that their rectify strategy were always better than the direct and recursive strategies[42]. This paper shows that the rectify strategy is very attractive for multi-step prediction tasks.

2.2.4 Multiple Output Strategy

The multiple output strategy involves developing one model that is capable of predicting the entire forecast sequence in a one-shot manner. In the case of predicting the temperature for the next two days, we would develop one model and use it to predict the next two days as one operation. For example:

$$prediction(t+1), prediction(t+2) = model(obs(t-1), obs(t-2), ..., obs(t-n)) \tag{2.7}$$

Multiple output models are more complex as they can learn the dependence structure between inputs and outputs as well as between outputs. Being more complex may mean that they are slower to train and require more data to avoid overfitting the problem.

2.3 Time Series

A time series is a sequence of observation of data points measured over a time interval. Time series data have a natural temporal ordering. This makes it distinct from common data problems, where there is no natural ordering of the observations, and from spatial data analysis, where the observations typically relate to geographic locations. Time series data can either be discrete observation spaced at defined interval, e.g. weekly share price, daily rainfall, or continuous observation made at every instance of time, e.g. lie detector. Time series are used in pattern recognition, statistics, signal processing, mathematical finance, weather forecasting, earthquake prediction, power-demand prediction and many more.

A time series is a sequence of observations taken sequentially in time. [5]

2.3.1 Time Series Forecasting as Supervised Learning

Time series forecasting can be framed as a supervised learning problem. This grants us access to the suite of standard linear and nonlinear machine learning algorithms. Before diving further into what this re-framing means, understanding what supervised learning is and how the foundation for all predictive modeling machine learning algorithms, is imperative.

Supervised Machine Learning

Supervised learning can be described as having input variable **(x)** and output variable **(y)**, and use an algorithm to learn the mapping function from the input to the output.

$$y = f(x) \tag{2.8}$$

The purpose is to approximate the real underlying mapping so well that when receiving new input data **(x)**, one is able to predict the output variable **(y)** for that data. The process of an algorithm learning from the training dataset can be thought of as a teacher supervising the learning process, which is where it get its name from, supervised learning. By knowing the correct answers, the algorithm makes predictions on the training data and is corrected by making updates. This learning stops when the algorithm has achieved an acceptable level of performance. Supervised learning can be divided further into classification and regression problems, and in this thesis we look into regression problems. A regression problem is when the output variable is a real value, e.g. "temperature" or "power-demand".

Sliding Window For Time Series Data

Now that we have established what supervised machine learning is, we can look into how time series data can be phrased as supervised learning. A time series dataset can be restructured to look like a supervised machine learning problem by using previous time steps as input variables and use the next time step as the output variable. Figure 2.2 shows this.

X	Y
time step 1	time step 2
⋮	⋮
time step n	time step n+1

Figure 2.2 Sliding window for time series data

Depending on the amount of input variables and output variables the sliding window can become increasingly more complex, which is where classical methods tends to not perform well. Machine learning methods tends to thrive where these classical methods fall, especially because of this additional complexity when dealing with multivariate time series. Even complex univariate time series can be too much for classical methods, which is again, where machine learning methods takes up the slack and out performs them.

2.3.2 Time Series Analysis vs. Time Series Forecasting

There are mainly two ways of approaching time series forecasting, whether its understanding a dataset or making predictions. Gaining understanding of the time series at hand, also known as time series analysis can be very helpful in making better predictions, but is not a necessity. Time series analysis can take a long time and may not always directly align with the desired outcome. In the book "Practical Timer Series Forecasting with R: A Hands-On Guide" page 18-19 they say[40]:

> *In descriptive modeling, or time series analysis, a time series is modeled to determine its components in terms of seasonal patterns, trends, relation to external factors, and the like. ... In contrast, time series forecasting uses the information in a time series (perhaps with additional information) to forecast future values of that series.*

Time Series Forecasting

The way time series forecasting differs from normal forecasting is that time series adds an explicit order dependence between the observations, a time dimension.

This dimension can both be a constraint and a something that provides a source additional information. This does not count for problems that are re-framed into supervised learning problem, mainly because you remove this dimension and can use evaluation metrics such as k-fold cross validation. If one still wants this dimension there are other options such as walk forward validation, which maintains this dimension. Forecasting is when models are trained on historical data and uses them to predict the future. An important characteristic in forecasting is that the future is completely unavailable and is only estimated from earlier observations. Time series forecasting models, like any other prediction model is determined on their ability in predicting the future, which often comes at the expense of understanding why a certain prediction was made or root behind the problem. Therefore the most important objective is to grasp a good understanding of what the main objective or goal is. Asking many questions may help narrow down the specifics of the problems that needs answering, some questions that might help can be:

1. How much data is available? If one has access to a large amount of data, then it would help with data analysis, model testing and tuning.
2. What is the time horizon of predictions that is required? Short, medium or long term?
3. Can forecasts be updated frequently over time or must they be made once and remain static? This depends on the different techniques and strategy used, nevertheless updating frequently as new data is available will benefit the accuracy of the model.

Time Series Analysis

> *The purpose of time series analysis is generally twofold: to understand or model the stochastic mechanisms that gives rise to an observed series and to predict or forecast the future values of a series based on the history of that series.*[41]

This quote gives a general explanation of why someone would perform a time series analysis. Time series analysis entails the construction of models that encapsulates or depicts the observed time series in order to extrapolate the fundamental reasons. I.e. looking for "why" the dataset is as it is. This can be done by dividing the data that is being observed into components. What this decomposition does is provide a better understanding of the dataset. These components can be divided into four parts. The first part, is the baseline value for the time series. The second part, is how the linear behavior of the time series is over time, whether its an increase or decrease. The third part, is the patterns or the cycling behavior over time, this can be optional as not every time series has recurrent pattern. The fourth and final part, can also be called the noise part, is the part of the time series that cannot be explained or is difficult to find reason behind. Even though not every time series

may have a trend or seasonality, when they occur they tend to be the main features of that time series. A quote that describes what was stated earlier says something similar:

> *The main features of many time series are trends and seasonal variations ... another important feature of most time series is that observations close together in time tend to be correlated (serially dependent).*[26]

Although, one decides to do a time series analysis and combine these parts, it does not necessarily mean that it would always be able to deduce something of value and create a model that would result in good performance, but may lead to other useful information, and might even lead to unexpected results unforeseen.

Univariate/Multivariate Time Series

There are different time series forecasting problems, this depends on the dataset. If the dataset only has a single variable observed at a time, such as power-demand for each hour, then its a univariate time series. If the dataset has two or more variables are observed at each time then its a multivariate time series. The main difference is that with more variables the complexity increases as well.

> *Multivariate time series analysis considers simultaneously multiple time series. ... It is, in general, much more complicated than univariate time series analysis.*[43]

2.4 Prediction methods

In this final section a short and informative description of the different prediction methods used in this thesis will be given. These methods consists of autoregressive, machine learning and deep learning. These short descriptions are given as preparation for the relative literature in the next chapter.

2.4.1 Autoregressive

Autoregressive (AR) models and processes operate under the premise that past values have an effect on current values, which makes the statistical technique popular for analyzing nature, economics, and other time-varying processes. The term *autoregressive* originates from the literature on time-series models where observations

from the previous time-steps are used to predict the value of the current time step [3]. Thus as one could have guessed, autoregressive models are used for time-series problems, and is a more traditional way in predicting future behavior based on past behavior, in comparison to the two next methods.

2.4.2 Machine Learning

There are a lot of different types of machine learning (ML) algorithms and techniques, but a general definition given by Dr. Yoshua Bengio, from Université de Montréal is:

> *Machine learning research is part of research on artificial intelligence, seeking to provide knowledge to computers through data, observations and interacting with the world. That acquired knowledge allows computers to correctly generalize to new settings.*[28]

The type of ML that will be used in this thesis is supervised regression ML. This can be defined as having all data labeled and the algorithms learn to predict the output from the input data, as well as the having the output variable being a real value, making it a regression problem. Generally, ML can learn more complex pattern and therefore can prove to be better in certain scenarios where we have higher complexity in the problem we are trying to solve.

2.4.3 Deep Learning

The concept of deep learning (DL) was proposed by Hinton et al. [14] in 2006, originated from the study of Artificial Neural Network(ANN). DL can be seen as a subset of ML, and both ML and DL a subset of artificial intelligence, but DL usually consists of more complexity or numerous layers of these algorithms. A good example for showing this is the Random Forest algorithm. It simply consists of many RTs (Regression trees) which are a ML algorithm, making it into a DL algorithm.

Even though DL is not the main focus in this thesis, having included DL algorithms such as LSTM and RNN that has not been used in this research field before, making them the novelty of this thesis.

2.4.4 Advantages and disadvantages

There are distinct advantages and disadvantages for which approach to be utilized. Whether it is a ML, DL or a regression based approach depends on the available resources, complexity, computational time and accuracy. Yet in general, ML and DL models seem to have an advantage over regression models for prediction tasks [48, 11, 1], mainly due to complexity and more data available to work with. Though in some cases a hybrid approach might be the solution [19] (ML and AR). It can also be defined as a complexity meter, where AR models has low complexity, ML models has medium complexity and DL models has high complexity. Therefore by identifying the complexity of the problem one can choose the appropriate method of choice. This is mainly generally speaking, and some types of problems can be solved with multiple or a combination of the methods. A recent study [24] evaluated and compared many classical, modern machine learning and deep learning methods on a large set of univariate time series prediction problems, and the results of the study suggests that simple classical methods such as Theta, ETS and ARIMA would outperform complex and sophisticated methods, such as Multilayer Perceptitrons (MLP) and long short-term memory (LSTM) network models.

Chapter 3

Relevant literature

3.1 K-Nearest Neighbor

K-Nearest Neighbor (KNN) is also well known in the ML community, and considered one of the simplest ML models. KNN stores all available cases and predicts the numerical target based on a similarity measure (i.e. distance function). The most common distance function is euclidean. In other words the algorithm computes the distance to every training example x, then out of those distances picks k closest neighbors, and take the mean of their output.

Despite being one of the simpler ML algorithms, KNN have shown to be both fast and effective in power demand prediction at charging stations[22]. The study compared four different algorithms, in which they ended up picking KNN for its significantly low computational requirements. The objectives of their study were to create a cell phone application, that would give waiting time estimation for charging at certain charging station to EV owners. Even though some of the other models proved to have better accuracy, they still ended up picking the KNN model. This was mainly due to the fact that the applications speed was more significant, than a better accuracy. In one of their later works, [23], they looked into how to improve their KNN model. They decided to change distance function (euclidean) to the dot product based dissimilarity. This proved out to be a better choice resulting in less prediction errors, than their previous distance function. Another interesting result in the study were that they tested three different approaches for parameter selection, namely, k-fold validation, time series cross validation and block validation. They found that all approaches led to similar behavior in the results, but since the block validation were less computationally expensive, it was the best choice. Another significant discovery using KNN were that k = 1 showed better performance, which gave them an indication that it was always better to look at the most similar event in the past and copy its future energy consumption values as the prediction. They also started using Weighted KNN to add weightings to the dissimilarity measure, to ensure that recent similar indices had more weight than older ones.

3.2 Support Vector Regression

Support Vector Regression (SVR) derives from the ML algorithm Support Vector Machine (SVM), which is mostly used for classification problems. This being said, SVR is still perceived as a highly effective algorithm. The goal of SVR is to try to fit the error within a certain threshold. This threshold is defined by something called a hyperplane (defined as a line to help predict continuous values). The hyperplane passes near each point such that they fall within a specified distance of the hyperplane. The hyperplane is constructed based on support vectors (the closest nodes to the hyperplane), thus creating a boundary which ignore the errors as long as they are less than the margin, but will not accept any deviation larger than this.

With SVR, one can achieve an excellent prediction accuracy as witnessed in a study [22], where they compared four different algorithms, resulting in SVR having one of the best accuracies. Even though SVR had a better accuracy than some of the other algorithms, it was rendered inappropriate and a mismatch for their objective, because of its computational cost. Chen et al. [6] entered a competition for finding a solution to load forecasting, where they had to predict the daily maximum load for the next 31 days. They proposed a SVR model and won the competition. Depending on different criteria, SVR can be the better algorithm.

3.3 Regression Trees

Regression Trees (RT) or its former name, decision trees (RT is its modern name) can be like most other ML algorithms for regression problems and classification problems (usually known by the name Classification and Regression Trees (CART)). RT can be pictured as a upside down tree(or roots), were the tree consists of decision nodes and leaf nodes. A decision node has two or more branches, and a leaf represents a decision. The topmost decision node is called root node. It breaks the dataset into smaller subsets, while at the same time incrementally developing a decision tree. Based on the input data the RT model will generate rules, which are used to make the predictions. These rules are a mapping from the root node to the leaf nodes one by one.

RT can prove to be effective and quite fast, as shown by Ruiz-Abell at el. [39], where they compare four different models that are based on RT. There were two methods primarily that showed decent results, namely, Random Forest (RF) and a boosting method called XGBoost. It were discovered that the latter method mentioned had a quite fast computation time. RF however is one of the most used algorithms, because of its simplicity. RF, simply worded, builds a collection of RTs and merges them together to get a more accurate and stable prediction. RF is slightly different from RT in the sense that it is not defined by rules. RF models are especially useful at handling high dimensional problems. Although, in most cases at the expense

of computational power. Another study, [11], where they used an ensemble of models to achieve better accuracy, they used eight different models including ML and regression models, where the RF and SVR had the best accuracy, thus gaining most weights in the ensemble models.

3.4 Auto regressive models

An autoregressive model is described as when a value from a time series is regressed on previous values from that same time series. When one combines an autoregressive (AR) and a moving average (MA) model one gets the ARMA regressive model, which is known for linking the present value of the time series to its past values as well as some past random error respectively. ARMA models when integrated with an additional differencing order d in order to remove the possible nonstationarities within the data are called ARIMA. These simple models are widely used for load forecasting [34], [7] and [30], and can be used to accurately predict hourly and peak loads when modified [2]. ARIMA is a popular and widely used statistical method for time series forecasting. ARIMA is an acronym that stands for AutoRegressive Integrated Moving Average.

3.5 Recurrent Neural Network

Recurrent Neural Network (RNN) is a type of ANN that has a recurring connection to itself. RNN is considered to be a very popular Deep Learning model, commonly tasked with time series prediction, image captioning and grammar learning. RNN uses recursion technique to build models. Simply put it is a kind of neural network, which will send current data back to itself. As a result it has a memory, and can "remember" the history data. Thus the data travels in both directions, which is different from the traditional way information flows in neural networks. RNN is known to have two common issues: exploding gradients (minor issue) and vanishing gradients (not easy to fix).

RNN has been proved to be very effective in plenty of other fields including music composition[9], automatic speech recognition[13], but there are very few (if at all) studies covering prediction of power demand of charging station with the use of RNN. In a comparison between a SVM and a RNN model for load prediction of non-residential buildings witnessed in one study[18]. The RNN model turned out to achieve a higher accuracy, than the SVM model, while also having a low computational demand.

3.5.1 Long Short-term Memory

In LSTM, the network is capable of forgetting (gating) previous information or remembering it, in both cases by altering weights. This effectively gives an LSTM both long-term and short-term memory and solves the vanishing gradient problem. LSTM can deal with sequences of hundreds of past inputs. Vanilla RNN can only memorize the short time series data. With the increasing amount of data and time steps, it will lose the important information of long term input, causing a vanishing gradient or exploding gradient problem. LSTM were therefore introduced to tackle this [15].

There are very few studies on LSTM power demand prediction, therefore including studies done on LSTM outside of this topic is needed. Bedi at el. [4] compares various combinations of RNN, LSTM with EMD (empirical mode decomposition) for electricity demand estimation. The study ended up discovering that the right combination that would out perform the rest is a EMD + LSTM combination. In another study where they compare the algorithms Bayesian Knowledge Tracing (BKT) models against vanilla RNNs and LSTM based models [20]. The study resulted in LSTM achieving the highest accuracy.

Chapter 4

Data preprocessing and analysis

Machine learning algorithms learn from data. It is critical that you feed them the right data for the problem you want to solve. Even if you have "good" data, you need to make sure that it is in a useful scale, format and even that meaningful features are included.

4.1 Data description

The data consists of two files. The two files (File1) and (File2) contains observations from two different routes in Norway. File1 contains about 45000 observations from nine different charging stations, while File2 contains about 25000 observations from eight different charging stations. Both share the file format seen in figure 4.1. Take note of File1 and File2 as those two names will be used for these two files throughout this thesis.

year	month	day	hour	weekDay	chargingStation	time	chargings	plugType
2016	1	1	14	5	37	1037	1	CCS
2016	1	1	15	5	55	1082	2	CHAdeMO
2016	1	1	16	5	55	2463	1	CHAdeMO
2016	1	1	17	5	55	3455	0	CHAdeMO
2016	1	2	10	6	55	393	1	CCS
2016	1	2	11	6	55	728	0	CCS
2016	1	2	13	6	55	1003	1	CCS
2016	1	2	14	6	35	693	1	CHAdeMO
2016	1	2	15	6	35	1321	2	CHAdeMO
2016	1	2	16	6	35	3316	1	CHAdeMO

Figure 4.1

As most raw data files, these two needed to be processed and adjusted for us to be able to use them, which we will go more in depth in the next section. Each station had one to two years of observations beginning between early 2016 to mid 2017 and ending early 2018. These stations were given an id in the file for anonymity. One thing to take notice of in the files is that they don't include the power-demand for each observation, but has the time spent charging in seconds, and the plug type used. With this one is able to estimate the maximum power-demand which we will go through in the next section.

Side note: File1 was given late June 2019, which made it difficult to identify different charging station types with only File2 amount of data observations.

4.2 Data preprocessing

4.2.1 Power-demand estimation

Data can come in different forms and scales, thus adjusting it to fit a certain goal is necessary. The data given by IFE did not include the actual power-demand as previously stated, but enough data included to deduce the max power-demand that were used. The data that were used to do this estimation were the plug type (i.e. AC/CC or CCS/CHAdeMO) and the time spent charging. As described in section 2.1.1, the plug types give the necessary information about the maximum output the plug can give, and thus the calculation of the maximum power-demand can be estimated and used. For 50 kw chargers (i.e. CCS/CHAdeMO) the kw per second is $\approx$ 0,0139, while for 22 kw chargers (i.e. AC/CC) is $\approx$ 0,0061. The equation used to estimate the maximum power-demand for each observation:

$$kWh = time * kW/s \tag{4.1}$$

Where kWh is the maximum kilowatt hours that plug at the charging station could have had that day, and where kW/s is the amount of kilowatt one gets (depending on the plug type used) per second. kilowatt per second is what you get when you take the kw of that plug type and dived it by amount of second in an hour, thus getting the two numbers presented earlier. The reason behind using kw/s is because the time observations in the dataset were in seconds.

4.2.2 Data adjusting

There are multiple data elements that had to be adjusted so that the ML model could use it efficiently. First and foremost the week day element had to be changed into a binary format. The reason a binary format is used instead of natural numbers, e.g. numbers 1-7, where 1 is Monday and 7 is Sunday. This is mainly due to the fact that machines do not comprehend that the distance between 7-1 is the same as any other distance to the adjacent numbers. This is solved by converting the week day into a binary format.

The observations were hourly, but when there were no one charging at the charging stations there where no observations thus adding this missing zero value to the data was needed. Therefore one had to iterate through the whole raw data file and add every single hour that were missing. This had to be done for us to be able to

know when there is an opportunity to charge the reserve batteries at the charging stations. The files that did all of these adjustments and preprocessing can be found in the appendix named preprocessingData.py and dataAnalysis.py.

Side note on the adjustment of the data: When we started testing the data we simplified the problem by adding all observation for a day into one observation. This was mainly due to the fact that the observations were hourly and there were a good amount of hours with no observations. However, this was not due to missing data as previously stated, but rather that there simply were no one charging at that hour. The data also had days with no observations, thus observations with zero power-demand had to be added. This would simplify the use of the data, by being able to shift the data n amounts of days to be used as the Y training set i.e. for evaluation of the model. Another way of approaching this problem which might prove to be better at predicting further down the time-steps, is to add every week or day into batches in the dataset, thus making each time-step a day's prediction consisting of 24 observations or 24 multiplied with 7 days of observations in one prediction. This might make the multi-step prediction easier, and might be ideal for future testing especially for the later time-steps in the prediction.

X	Y
day 1	day n
⋮	⋮
day n	day n+n

Figure 4.2 Shift training set X with n days

4.3 Data cleaning

4.3.1 Data screening

The accuracy of the ML algorithms depends on the quality of the data (in most cases). If the data consists of missing data and outliers, the models might be inefficient, rendering them useless in a real world scenario. Therefore inspecting the data for errors, and correct them prior to the testing or data inspection will significantly improve the results in the end. Another reason for data screening is to aid the inspection of data, thus ease the process of identifying differentiates in charging stations.

4.3.2 Handling missing data

In ML there are certain approaches to deal with errors in the data collected. One of the errors is that they are not there, i.e. missing data. Missing data is one of

the greatest challenges analysts deal with, and hence there is a variety in both complexity and strategies in dealing with missing data. Missing data imputation can very easily be done by replacing the missing numerical data, with mean, median or mode of the feature. This strategy is one of the simpler methods in dealing with missing data, and usually used in social science because of its simplicity. This method is not viewed as the best option as it may add variance and bias to the data.

ML methods for estimating missing data is another approach to deal with missing data. It usually consists of creating a predictive model to estimate values that will replace the missing data using information from the dataset. It is witnessed in this study [17], that ML methods are more suited for imputation of missing data. The study compares statistical methods (e.g. mean imputation), with ML methods in imputation of missing data. The study also mentions that the amount of missing data should be considered, when evaluating the method of choice. If the quantity of missing data is large, then the use of a sophisticated procedure may be the best choice, otherwise using a less time consuming method might be more appropriate.

As mentioned earlier the data had missing observations, and therefore had to be added for use to be able to know when it would be possible to charge the reserve batteries as well as how much kWh were available to charge the battery.

4.3.3 Handling outliers

Outliers are extreme values that fall a long way outside of the other observations. Outliers in data can mislead the training process of ML algorithms, resulting in less accurate models and longer training times. There are many methods existing for outlier detection. Extreme Value Analysis (EVA) is one of these methods. One way of utilizing EVA is to visualize the data using scatter plots or histograms and look for extreme values. Another way is to filter out outliers "candidates" from the training set and assess the performance of the model. Another method is to use ML models that are robust to outliers.

In figure 4.3 some of the stations have random outliers that can create noise in the training process of a model, or if able to learn from them be able to predict these huge peak demands making it possible to adjust for such days. Therefore in the future when doing a more in depth research of one model, maybe consider removing or marking them as outliers in the data files.

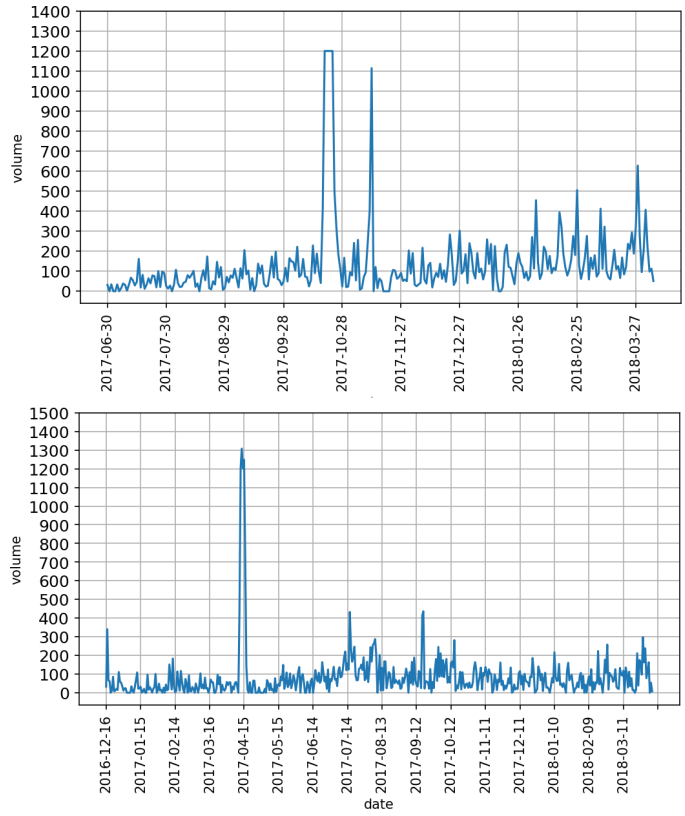


Figure 4.3 EV charging station with outliers

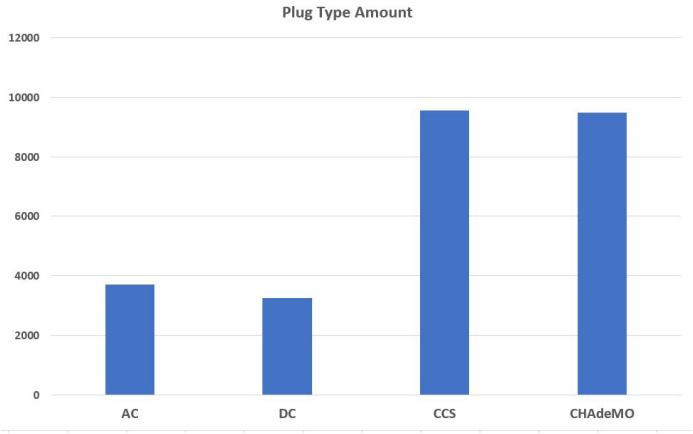


Figure 4.4

4.4 Data analysis

4.4.1 Plug type amount in data

Figure 4.4 clearly shows that fast chargers at EV charging stations have a dominance in usage, with twice as many observations of fast chargers recorded in the data from file2. This is also to be expected insight of people not wanting to wait for more than 20 minutes to charge their vehicle.

4.4.2 hourly power-demand pattern

The figures 4.5 and 4.6 shows the hourly power-demand for both routes. File1's peak starts at 12:00 and end at 20:00. There are few charging observations between 23:00 and 08:00, but the usual consumption starts at 09:00 and ends at around 22:00. File2 a little bit different pattern insight of having a lot less power consumption in total. The average hourly consumption of the day for File1 is $\approx 6.2814 maxkWh$, and the peak demand varies between 200-500 max kWh. In comparison to File1, File2 with 20 000 less observations has a much smaller peak. The peak demand varies between 50-200 max kWh, and the average hourly consumption in File2 is $\approx 4.3696 maxkWh$.

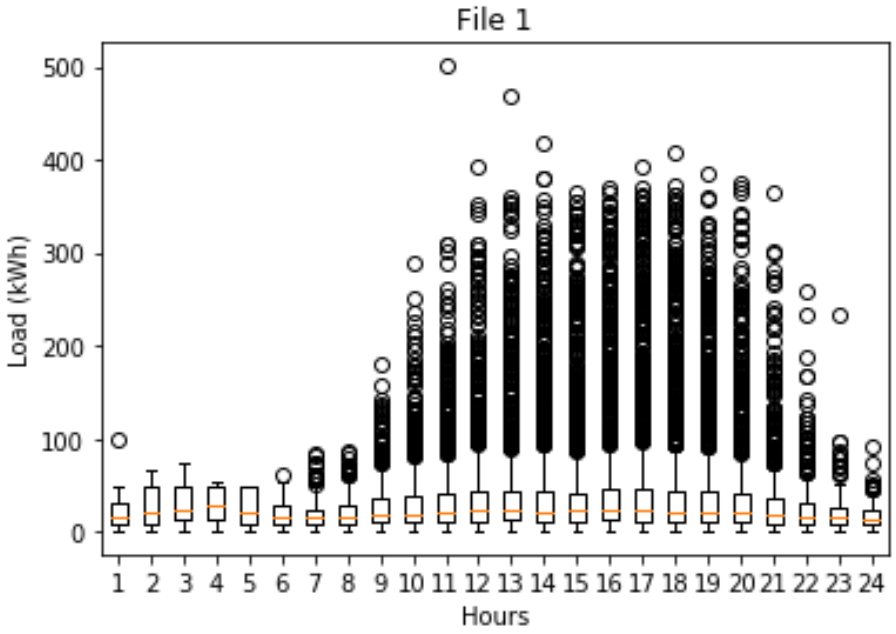


Figure 4.5 Hourly power pattern of all the charging stations in File1

4.4.3 Daily power-demand pattern

Both data files has the same types of peak days. Figure of File 1 4.7 shows more data, because it has more data 45000 observations, compared to 25000 observations in File 2. It shows Friday(5) and Sunday(7) to have the largest peaks in the data and also shows that the average is higher on those two days. In File 2, shown in the figure 4.8, one sees the exact same pattern with the peak loads being mainly on these two days. This back the finding of an analysis done by another scientist at IFE discussed further down.

Adding a binary for every Friday and Sunday might help the model adjust to these peaks, insight of these two days reaching higher peaks than the other days of the week. We were not able to identify the reasons for the other random peaks in the other week days. These outliers does not seem to have any pattern to them, all

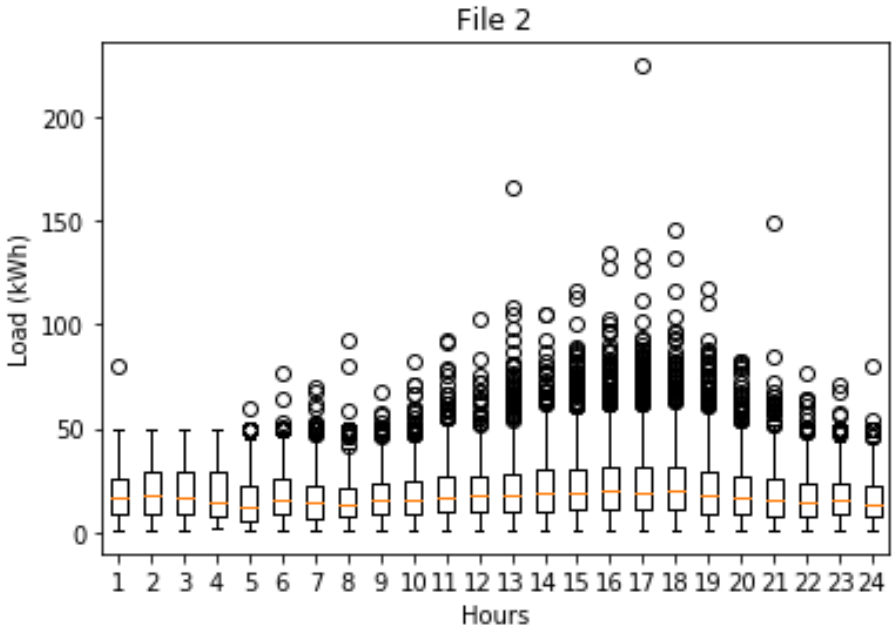


Figure 4.6 Hourly power pattern of all the charging stations in File2

happening on random dates and random days with no correlation. Looking into this might help future research identify special case days, thus helping the model more.

In a larger analysis done by some scientists at IFE, made on the data used in this thesis, plus some more, they were able to identify, Fridays, Sundays, and days near vacations to be the peak power-demand days. The figure 4.9 shows how on these days in particular that it is a larger demand, than on regular days. This further backs the special day binary that will be discussed in the section Extra data included.

4.4.4 Extra data included

Parameters included as part of the Input data. These parameters were introduced as candidates for parameters that can help with the models accuracy.

Temperature Insight of there being very little research done on how temperature impact EV's battery performance. Patten at el.[31] did a study where they documented a modified Toyata Pruis with a 5 kilowatt-hour(kWh) plug-in battery for one year, mainly to determine how various temperature conditions affected vehicle performance. They compared both fuel economy and pure electrical efficiency. The study resulted in that the fuel economy has a positive relationship with ambient temperature until approximately 283 K where the efficiency begins to level off. Electrical performance has a positive linear relationship with ambient temperature.

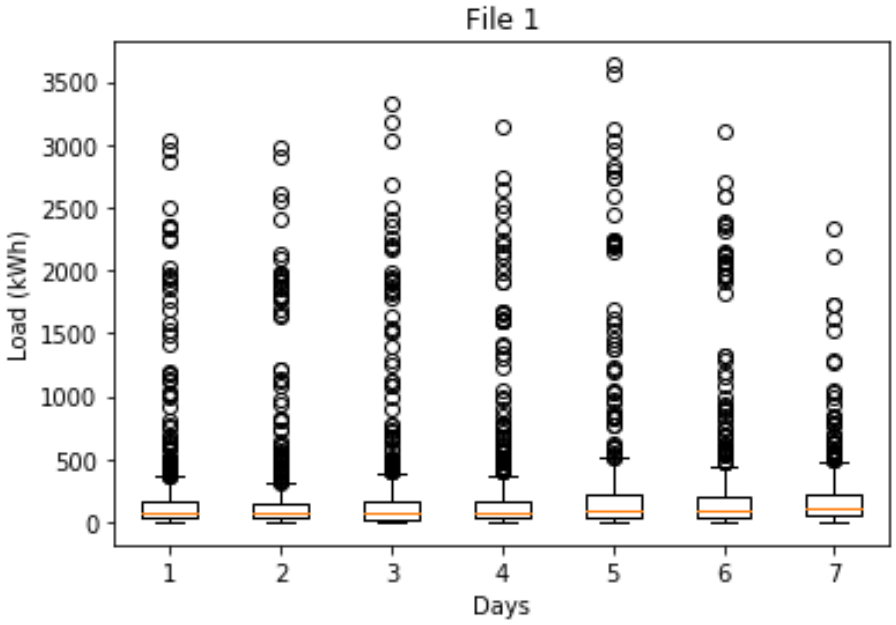


Figure 4.7 Daily power pattern for all charging stations available (1 is Monday)

In this thesis it will be tested with a constant temperature for summer(April-September) (10 °C) and a constant temperature for winter(October-March) (-10 °C). This is mainly due to the fact that it would take a lot of time to get the temperature for the available data, but in the future getting real temperature data for all hourly observations would be an interesting finding. The fact that we are using constant temperature parameters has a big downside to it, this is mainly due to the fact that we wont know how the temperature effects EVs power-demand pattern in different seasons of the year. Only having a constant will not only be very inaccurate when it comes actually picking up the pattern of how temperature effects EVs power-demand. In the results we will comment on this, but also have in mind that this wont be accurate and can't be taken literally. It will still give an indication of how temperatures can effect power-demand patterns of EVs, because it is colder on the winter than the summer.

Peak days binary As shown in the figure 3.3, Thursday and Sunday usually have higher power-demand, compared to the other days thus giving these two days a binary variable to help differentiate between the days and increasing the accuracy of the models.

Week Day binary A binary week day number to help the model classify the different days to a variable.

Previous KWh Including the KWh of the previous observations might help train the model. The parameters that will be added to the tests are, the previous 12 hours. Before switching to hourly observations, the prediction used to include the previous 7 days of kWh as parameters.

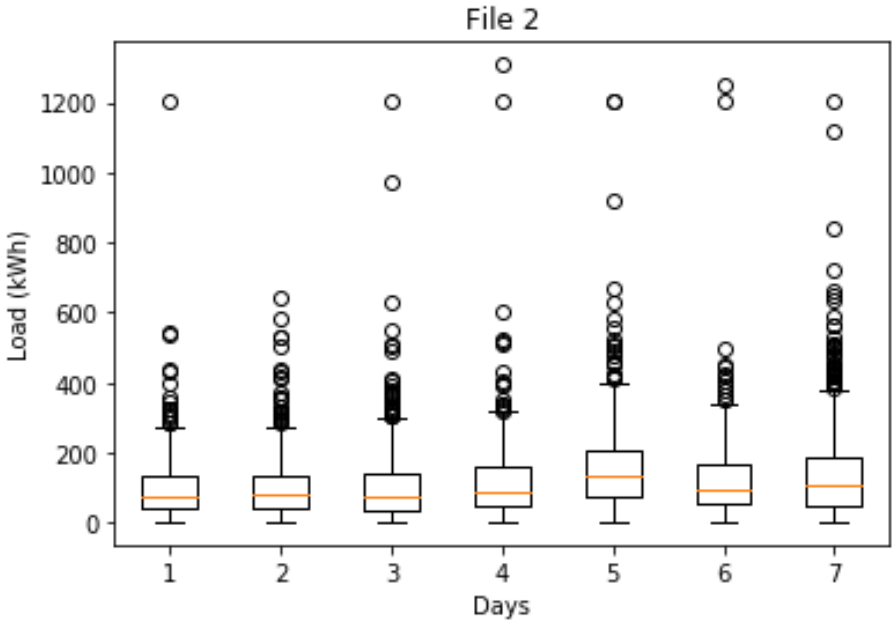


Figure 4.8 Daily power pattern for all charging stations available (1 is Monday)

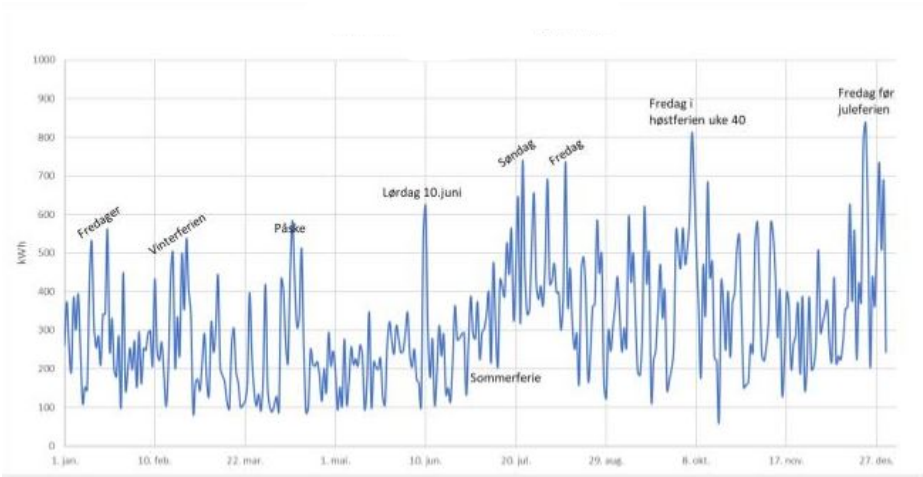


Figure 4.9 Peak analysis

Previous predictions Using previous predictions as an input in the training of the models is something that will be used when testing the recursive and hybrid multi-step strategies.

4.4.5 Station categorization

One of the goals that we wanted to achieve from this master thesis was to identify or categorize 5 different EV charging stations. This would then benefit the evaluation of each model when comparing them to the different models. By understanding whether certain models are better for certain types of charging stations, thereby

identifying which model might work better with a certain charging station type. As mentioned in section 2.1.2, this will increase the accuracy of the predictions. Unfortunately, this was not possible due to the similarity in the available data. There were some recurring pattern in some of the charging stations, but not anything significant. A pattern that was commonly seen in most of the charging stations, were a upwards trend in the use of charging stations. This is due to the increase in popularity in EV consumers. Some of the charging stations with the least action usually had large peak loads, which can be identified as anomalies in the sense that it was irregular power-demand consumption at that charging station.

Chapter 5

Prediction models

In this chapter the models that are going to get compared are presented, as well as the programming tools and libraries used for their implementation. Insight of this being a comparison of six different algorithms the models will be standard right out the package models with some small feature tweaks. This process was done by picking one random model of the five algorithms discussed in the relative literature and fine tuning the input data without adjusting too much on the model itself but rather putting the focus on the parameters given to the model. Identifying potential parameters will give a larger and wider testing ground for when comparing all methods.

5.1 Models and implementation

Insight of the objective for this thesis being a comparison between multiple methods thereby identifying which method works best in which aspect, to elaborate more by testing these methods and inspecting the strengths of each method when using certain inputs or which time step the prediction is for. How each of the models will be tested will be described further down and how they are implemented. There were originally these six methods that were going to be implemented and compared: LSTM (Long Short-term Memory), KNN (k-Nearest Neighboor), SVR (Support Vector Regression), RT (Regression Tree), AR (Autoregressive), and RNN (Recurrent Neural Network).

As previously stated, the AR model is only here to make sure that we even need to use ML models in the first place, thus having a more tradition model in the comparison helps us identify the need for the use of ML models or not. The of which the master thesis is under has very little to no research on how deep learning models perform in power-demand predictions. These, [29],[21], are some other sources where they explain how to develop both multi-step LSTM power-demand prediction models and regular LSTM power-demand predictions models. LSTM is a further developed version of RNN and both are deep learning models, making deep learning models the novelty of this thesis.

Implementation process: Firstly, experimented with parameters and strategies on one of the models, the choice of model is not important in this stage (A RT model were used here). After identifying potential parameters and the input data had been preprocessed the rest of the models were implemented. The models were not based on any of the relative work models, instead they are a implementation based of the algorithms that are available from the external python libraries described in subsection 5.1.1 Programming tools and libraries.

Modal RNN The implementation uses the external library tensorflow, where it reads in data using a function placed in the file Utils.py to read the input data. It then calls on Sequestial() to initialize the model. The model consists of three layers, the first layer being the simpleRNN layer which consists of 200 neurons with the activation function "relu", and the input shape. The second layer is a dense layer that consists of 100 neurons also with the "relu" activation function. The last layer is also a dense layer with only 1 neuron as we only want to predict one thing, the power-demand. The model is is compiled with the loss metric "mse", the optimizer "adam" and the metrics "accuracy".

Modal RT The implementation uses sklearn another external library, which offers a variety of options and settings for the model. In this scenario we used the straight out of the box model where there were no setting tweaks. The function has a recursive setup where the same model is used for each prediction time-step. The RT model sklearn library provides is very easily usable, which made the testing and implementation of the model enjoyable. The model implementation was straight forward and easy. Compared to the other models and libraries. Even though the library provides an easy to use model, it is still fully possible to specify settings and options for the model, thus making it a good candidate for furture development.

Modal KNN This model is also provided through the external library sklearn. The model is initialized at the beginning by calling neighborsRegressor() function imported from sklearn. The number of neighbors k can be determined with the use of a function called GridSearchCV and running the model with different k values ranging from 1-N. From earlier studies [22], discussed in the relative work chapter (3), k = 1 and is explained in that section of chapter 3. However, this did not prove to be the working in our case, but the larger the number the higher the accuracy the model got. The experimenting started with K = 24 and continued up to K = 500. The accuracy continues to improve, there were not a significant increase in accuracy after K = 200. We ended up having K = 150, since there were not a need for squeezing the model for more accuracy. Another adjustment to the model was picking the the weighted function 'distance', which has the description: "weight points by the inverse of their distance. in this case, closer neighbors of a query point will have a greater influence than neighbors which are further away.". This makes sense when predicting something in the near future, but the further the prediction steps are the less usable this weighted function is applicable, therefore adjusting the hyper parameters of KNN throughout the testing or predicting process would be the most beneficial way in using the model.

Modal SVR Again, it is also provided through the external library sklearn. The implementation consists of very few steps when it comes to the algorithm provided by sklearn. The model is initialized with a kernal set to "rdf", gamma set to auto and epsilon set to 0.01. This is a setup was found through experimentation and examples found on sklearn homepage.

Modal LSTM Both RNN and LSTM are models supplied by the external library keras who is built on tensorflow. The implementation of both models are almost exactly the same.

Modal AR The model was added to evaluate whether using a ML model is necessary and instead use a more traditional model. In our case this model is provided by statsmodels, and the model is a simple AR model. It is to specify dates, features and whatnot, but the model seemed good straight out of box. When predicting a start index and end index needs to be specify, this specifies at which time step to start and end the prediction.

Utils is a file including the necessary methods for making the reading of the data, printing of the graphs, and calculation of the evaluation metrics possible.

5.1.1 File structure & Designed functions

In figure 5.1 the file structure for the project is displayed. The input data is preprocessed and placed in data folder which contains the input data as both daily format and hourly format. The raw data can be found in the files: File1.xlsx and File2.xlsx.

Implementation notes

While developing the first ML model and testing different parameters and their formats, the four different multi-step strategies were tested as well and researched. What was deduced were that not every ML model would fit with the different strategies and that these strategies need to be used only on certain models where the algorithm itself does not get impacted by the strategy. E.g. SVR has only one output, therefore using it around a multiple output strategy for multi-step prediction would not be very beneficial. Another example would be RNN, using RNN in a Hybrid (RECTIFY) multi-step strategy would render the RNN's algorithm functionality useless. RNN uses its old inputs as a hidden state with the new input the next time it learns, therefore removing the model's bias by using a new one would make the RNN into a feed forward network(basically an ANN).

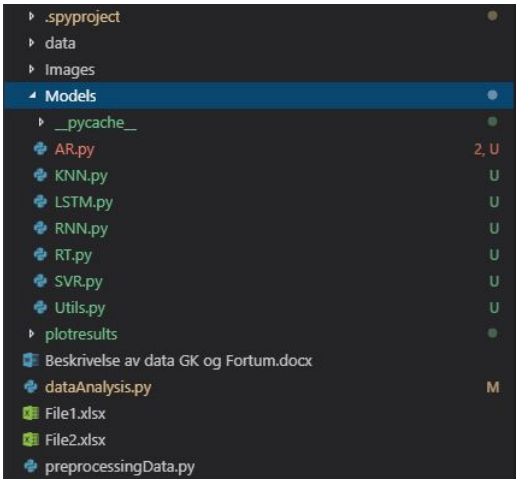


Figure 5.1 File Structure

5.1.2 Programming tools and libraries

To implement the different models some external Python libraries were used:

- **Keras** is a high-level neural networks API, capable of running on top of TensorFLow. Makes creating neural network models simple and runs seamlessly on CPU and GPU.
- **TensorFlow** is an end-to-end open source platform for machine learning. It has a comprehensive, flexible ecosystem of tools, libraries and community resources that lets researchers push the state-of-the-art in ML and developers easily build and deploy ML powered applications.
- **Scikit-learn/sklearn** are tools for data mining and data analysis. Mainly used to help create all other ML models beside the RNN model.
- **Pandas** is a library providing high-performance, easy-to-use data structures and data analysis tools.
- **Numpy** is a well-known library for numerical computation.

5.2 Tests

In this section the strategies and evaluation metrics used for the testing will be presented, additionally an explanation for why these were chosen.

5.2.1 Evaluation Metrics

How do we know how good a given model is?

There are evaluation metrics and strategies that helps us measure the accuracy of the models that we are testing. What will be used in this thesis is something called k-fold cross validation (which will be explained later on), and the evaluation metric RMSE. The reason behind using this combination for our evaluation of the different models is we need something to evaluate the accuracy of our models, but at the same time we need something that can help us with making sure that it was not based on certain part of the data, thereby having the cross validation technique and the RMSE metric, we will be able to deduce with more confident how well the models can predict the future. Before going any further into how this is done, giving a definition for these strategies, metrics and techniques is needed.

Mean squared error is a common way of assessing the quality of mapping of inputs to outputs. MSE is defined by Eq. 5.1

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(\hat{Y}_i - Y_i)^2 \tag{5.1}$$

Root Mean Squared Error is the root of MSE. It is a popular formula to measure the error rate of a regression model. However, it can only be compared between models whose errors are measured in the same units. Thus RMSE works for our comparison insight of the prediction being of the maximum kWh for a charging station. The data is scaled (with function MinMaxScaler() from sklearn) between 0-1, thus when the RMSE is given it is a percentage of error, e.g. the RMSE 0.12 is equal to 12 % RMSE. RMSE is defined by the Eq. 5.2

$$RMSE = \sqrt{MSE} = \sqrt{\frac{\frac{1}{n}\sum_{i=1}^{n}(\hat{Y}_i - Y_i)^2}{n}} \tag{5.2}$$

Modal scoring and error estimation for different time-step

Since the problem is a multi-step problem (see the background chapter section 2.2), where the prediction is for the week ahead, i.e. the next 7 days, the prediction will then be comprised of the next seven day's RMSE values, twenty-four for each day(hourly observations). It is common with multi-step prediction problems to evaluate each predicted time step separately. This is helpful for a few reasons:

- To comment on the skill at a specific lead time (e.g. +1 day vs +3 days).
- To contrast models based on their skills at different lead times (e.g. models good at +1 day vs models at days +5).

The units of the total power are kilowatts and it would be useful to have an error metric that was also in the same unit. Root Mean Squared Error (RMSE) fit this bill,

which is one of the reasons it is so commonly used. The RMSE mean/average is the arithmetic mean, i.e. the sum of all numbers divided by the number of numbers added together.

Where:
$\hat{Y}$: Vector of n predictions,
Y: Vector of n observed values

K-fold Cross Validation

There are plenty of cross-validation methods out there, even a train/test split can be identified as the smallest cross validation method, but the reason for using one is to thoroughly test the model with all parts of the data, thus reducing the biased of the models results. The results of k-fold cross validation summarizes the with the mean of the model skill scores. The k value needs to be carefully chosen, usually done through experimentation. Three common tactics are: *Representative*, the value for k is chosen such that each train/test group of data samples is large enough to be statistically representative of the broader dataset. *k=10*, the value for k is fixed to 10, a value that has been found to generally result in a model skill estimate with low bias and modest variance. Lastly *k=n*, the value for k is fixed to n, where n is the size of the dataset to give each test sample an opportunity to be used in the hold out dataset. This approach is called leave-one-out cross-validation.

> *The choice of k is usually 5 or 10, but there is no formal rule. As k gets larger, the difference in size between the training set and the re-sampling subsets gets smaller. As this difference decreases, the bias of the technique becomes smaller*[25]

The k-fold cross validation function that were used in the testing were provided from Scikit-learn/sklearn called KFold. Their description is as follows: Provides train/test indices to split data in train/test sets. Split dataset into k consecutive folds (without shuffling by default). The value k was set to 5 in our tests, insight of there were insignificant differences between 5 and 10 folds through experimentation while setting up the tests.

For future reference one should consider using a walk forward validation in testing the model further. With Walk forward validation, a model can be updated each time one receives new data. Therefore making this approach very interesting when real-time learning is in the picture.

Bias Vs. Variance

The ultimate goal of any ML model is to learn from examples and generalize some degree of knowledge regarding the task we're training it to perfrom. Some ML models provide the framework for generalization by suggesting the underlying structure of that knowledge. For example, a linear regression model imposes a framework to learn linear relationships between the information we feed it. However, sometimes we provide a model with too much pre-built structure that we limit the model's ability to learn from the examples - such as the case where we train a linear model on a exponential dataset. In this case, our model is biased by the pre-imposed structure and relationships.

Therefore models with high bias pays little attention to the data presented; also known as **underfitting**. See figure 5.2 for example of underfitting.This can also occur if the model is trying to solve a task without all the necessary information present.

The opposite as previously mentioned in chapter 4, is overfitting. This occurs when the model we train learns too much from the training data. This happens because the model captures the noise of the data in addition to the signal. This can cause wild fluctuations in the model making the prediction far from the true trend, models with overfitting tendencies has something called high variance. See figure 5.2 for example of overfitting.

In summery, high bias makes a model more limited when it comes to learning the true trend, and underfits the data. While a model with high variance learns too much from the training data and overfits the data. The perfect model is something in between these two.

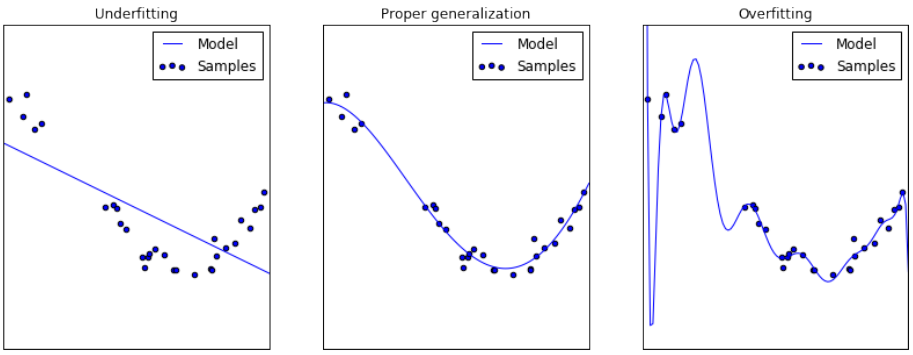


Figure 5.2 Proper generalization

5.2.2 Test implementation

This subsection goes through the various implementations of the three tests that will be discussed in the next chapter.

AR Vs. ML testing

This test is a simpler version of the next test, where we use a combination of sliding window technique and the recursive multi-step prediction strategy. The function is called ARVsMl(), which can be found in Utils.py under the folder utils. The function starts by reading the data, and uses only the parameter X1(max kWh), due to the restrictions that comes with using AR models, primarily that it only can handle univariate data input. After reading the data, it initilizates the models, which then brings us to the sliding window process and splitting the data into training and test datasets. This is a process where it trains and tests the models for every 24 observation, thus providing the RMSE value of the models ability in predicting that far in the future. We continue this process until it reaches the 168 observation, i.e. the 7th day, which then plots a graph of its ability to predict the data 168 observations into the future. The data used is a charging station from route A with ID 160.

Testing route A and B, & Parameter testing

The test implementation was created in a very dynamic way for the purpose of testing multiple algorithms similarly and without difficulty. The figure 5.3 shows the data flow of how each model were tested. Firstly each model has its own file, here the MultiStepPrediction() function which is where the model will be tested is called. Every model file has to provide a model initialization function and a model training and predicting function. For more specifics about what is needed to provide in the model files find model files in the appendices. After the MultiStepPrediction() is called, the readData() function which returns the data that will be used in the testing. This data is read from files that are preprocessed and ready to use. Now that the data has been provided, the preprocessing for the test get started by putting a side the amount of test data specified in the model file. By default this is set to one week because the goal of the thesis is to be able to predict the power-demand for the next week. This lives us with two sets one training data set and one test set with only one week of data. The training set is then further used into the k-fold function provided by Scikit-learn/sklearn, which returns two lists of indexes based of off the training data, and again, one training index list and test set. An illustration of how this works can be seen in figure 5.4. For each iteration of the k-folds the multi-step prediction is done where the sliding window tactic is performed. The training and prediction is done for each time step, and the time steps being 1-7 days. Insight of using hourly observations the training is done on every 24Th observation. This repeats until the last iteration of the k-fold is done. The whole process provides rmse measurements based on the current prediction, then the average for the whole week/k-fold iteration, and then finally the overall average for all the k-fold iterations.

The parameter testing uses the exact same test setup, except by running the test with the parameters under inspection, and using k = 4 in the k-fold cross validation.

The last thing to mention is that all the tests uses a combination of direct and recursive multi-step strategy. Some of the other multi-step strategies mentioned earlier in this thesis is more complicated to implement (recursive, RECTIFY and multioutput), which made the decision of using the recursive strategy easier, except for the direct multi-step strategy which would be a recommended to use in future research and implementations.

The last thing to mention is that all the tests uses a combination of direct and recursive multi-step strategy, it somewhat is a RECTIFY strategy, except that the previous predictions are not used in the prediction, but the same model is. This is mainly due to the fact that not every model can be easily used with some of the strategies, thus we decided to create a strategy that can be easily used with every model tested. This is a combination of direct, where we don't use previous predictions, and also use the same model. For future research a more direct strategy might prove to be more helpful against removing the models bias at predicting new time steps.

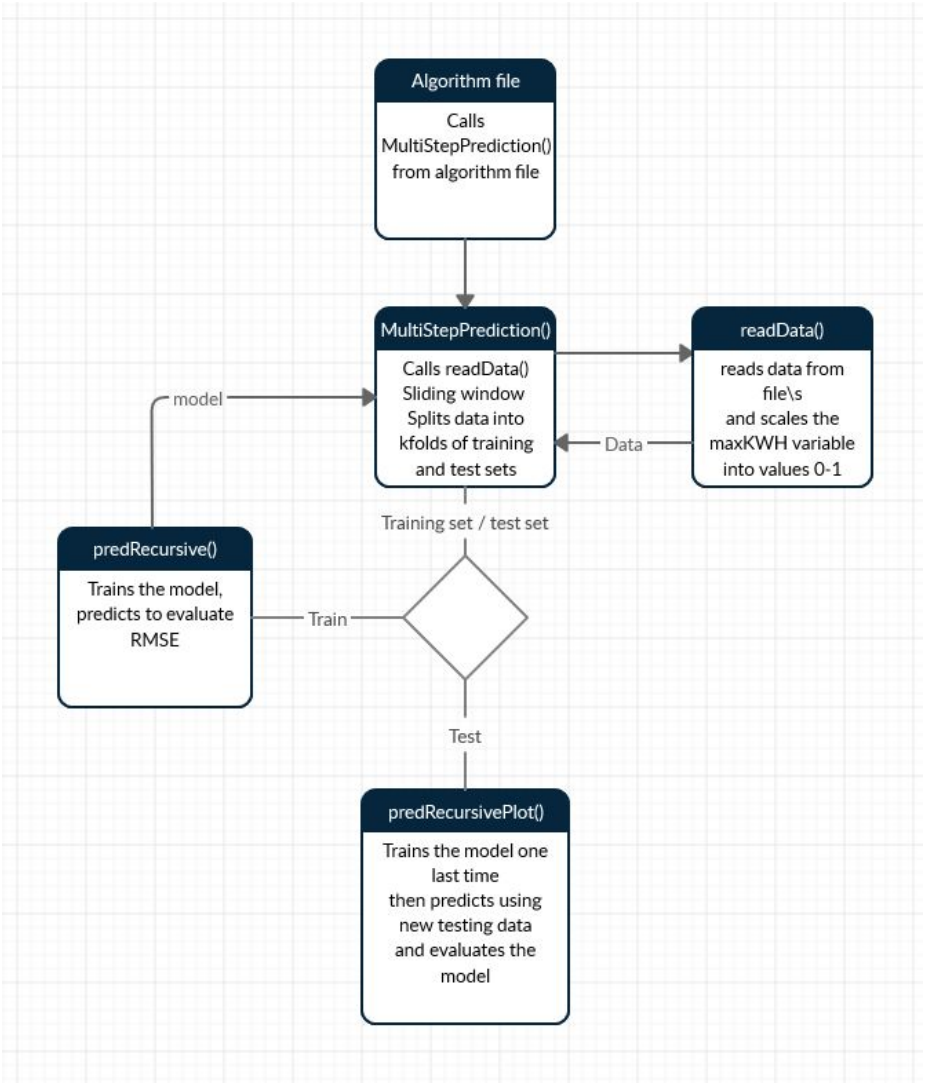


Figure 5.3 Data-flow diagram of test implementation

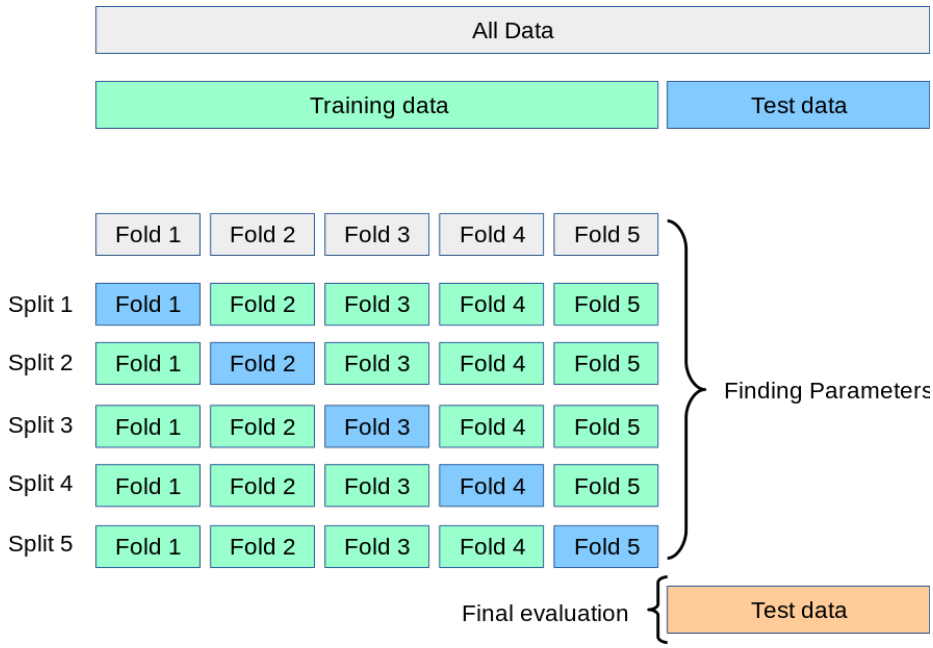


Figure 5.4 k-fold Cross Validation Illustration

5.2.3 Parameter Selection

The parameters used when testing consist of:

- Current kWh **(X1)**
- Week day binary **(X2)**
- Peak day binary **(X3)**
- Temperature **(X4)**
- KWh of the X previous hours **(X5)**

There are three main tests, first test will only include X1, the second test includes the parameters X1-X4, and the third and last test includes all the parameters one by one to show their influence and effect on each model, except AR. The reason behind not including X5 in the first test, is as will be explained in the next chapter some restrictions with using the AR model and it only taking univariate input data. As for the second test was primarily because of the amount of time it would take to generate the extra hours for each charging station and then do the prediction. Even without adding the X previous hours it took up to two hours for some of the models to finish the test. Therefore it will only be included in the parameter testing, where the amount of data used is limited to one charging station and where one iteration of the testing will include X = 24 previous kWh observations, and another iteration for X = 168. This will be enough to deduce if there is a need for adding X amount of previous hours, and the amount of previous hours is better. If we see a decrease in RMSE with more previous hours, then we'll know that there might

be a chance that adding more previous hours to the prediction might increase the models accuracy, but at the expense of computational time.

Chapter 6

Results and discussion

This chapter presents the results and findings from the tests performed throughout this thesis. The implications of the results are also discussed.

6.1 Tests

There were three main tests used in this thesis. Some consisted of multiple smaller tests, but in total three main objectives were tested. The first was whether the traditional method, namely the use of AR models, was sufficient for predicting power demand. The second was whether there was a difference in predicting the power-demand for high-activity versus low-activity charging stations. The third was how the parameters experimented with and found throughout this thesis affected the different models.

6.1.1 Conducting the tests

As mentioned, we wanted to test three main objectives. The first was to test traditional methods such as AR models compared to ML models. As stated in the previous chapter, the AR model is a simple AR model provided by the external python library statsmodels. Hence, this choice of model comes with some restrictions. One is that the model can only handle univariate input data; therefore, X1 (max kWh) was the only parameter that was used in this first test.

Another restriction is that AR cannot be used in k-fold cross validation. The AR model is intended for time-series problems, where the time dimension has a large role in the predictions, and this is how AR models work in general. One could use a technique called "walk forward" validation as a replacement for k-fold cross validation, as stated in Chapter 5, but that process was not used in this study. Instead we simply used an RMSE score from a regular train/test split. This test provided insight into whether a simple classical or traditional method was sufficient

to predict the power demand for EV charging stations. The data from one charging station on Route A, with the ID 160, were used.

The second objective was to test the models' abilities to predict the power-demand for charging stations with high versus low activity. This was tested by using the two routes given as data; file1 and its charging stations are referred to as "Route A" and file2 and its content are referred to as "Route B". Both these files are analysed and presented in Chapter 4 and appear in the Git repository.

Many discoveries can be uncovered through this test, but the main objective was to see whether there was a difference in predicting the power demand for the two routes. As a reminder, the difference between the routes was their hourly power-demand usage (see Chapter 4). Route A had an hourly power demand of ≈ 6.2814 max kWh, which peaked between 200–500 max kWh, whereas Route B only had ≈ 4.3696 max kWh and peaked between 50–200 max kWh. However, not all the charging stations in both routes were completely similar to each other; some charging stations, especially in Route B, showed periods with no activity at all. More information on which charging stations were used and what data were used appears in the subsections about each test.

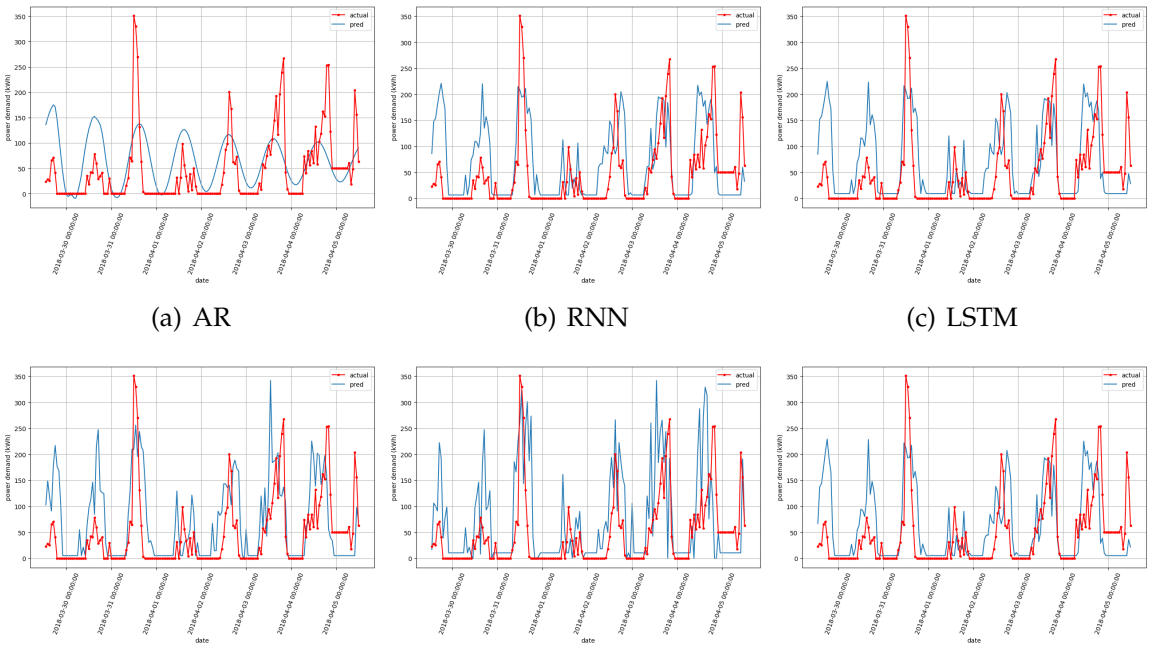
After having tested and compared the models on different routes, the parameter testing was conducted. First, a test with only **X1 (max kWh)** parameter was used to show how the models performed without other parameters. Then each parameter was tested with **X1**, for example, **X1 & X2 (week day binary)**, **X1 & X3 (peak day binary)** and so forth. This helped illustrate how each parameter affected the results and whether the model achieved greater accuracy with the extra parameter. How the parameters were chosen is explained in Chapter 4 (subsection 4.4.4).

The most important aspect of the results was the overall prediction ability of the model. The next was how well the model predicted each time step. Lastly, the computational time of the models was considered; however, this aspect did not influence the evaluation strongly, mainly because the tests were conducted on hardware that was probably slower than any used in a real-life context. Nonetheless, insight regarding the computational time of models can help future researchers to make sound decisions about which model would benefit them, according to different needs for different projects.

The figures 6.1-6.7 presents the prediction in blue and the actual data in red. All the plots illustrate the validation test prediction, which is a whole week and always the last week in the data used.

6.1.2 Comparing autoregressive and machine learning models

As previously stated, the AR model had some restrictions. Therefore, a separate test was conducted to include only X1(max kWh), to gain insight on how well it



(a) AR (b) RNN (c) LSTM

(d) KNN (e) RT (f) SVR

Figure 6.1 Test results for AR vs ML

compared to the other ML models in this particular scenario. There might be a theoretically superior model, but this might prove otherwise in practice, possibly because of the specifics of the problem being solved or the trajectory or aim of the objective being dealt with. Another factor might be the developer's ability to make a certain model work. Insight on how the different models operated with the specific data might prove beneficial for future research.

Table 6.2 shows the RMSE results of the six models, and Figure 6.1 shows the prediction of the 168th time step. The data used were from a charging station in Route A (ID 160), which all models seemed to be able to predict well. The RMSE values show that RNN and LSTM had the best overall RMSE of around 0.15. The AR model was in close third place; this was an unexpected insight as it is the simplest AR model offered by statsmodel.

All models seemed to show less accurate plots compared with the numbers presented in Table 6.2. This might be caused by zero values, as anticipated in Chapter 4. Regarding whether a more traditional or a modern method is most suitable depends on the task. If the task is simple and can be done in a straightforward fashion, and the problem is not complex, a more traditional method might work; RT or KNN can also be used as they are easy to implement and straightforward.

If the problem is larger and there are more data to use, a more complex method like RNN or LSTM, or any other sophisticated method, might help. Many research papers have stated this conclusion as well, as discussed earlier in this thesis. There is no superior model or method; rather, adjusting to the needs of a certain problem or situation is necessary. Overall, in this particular test RNN achieved the best

results. However, the results did not favor a single model; they all had about 0.15 RMSE value, except RT, which attained slightly over 0.18 RMSE value. For future recommendation, using a walk-forward validation combined with RMSE is highly recommended, because walk-forward validation fits with the use of new data and it is easy to add new data. The method also observes the time dimension in the data while using the data efficiently.

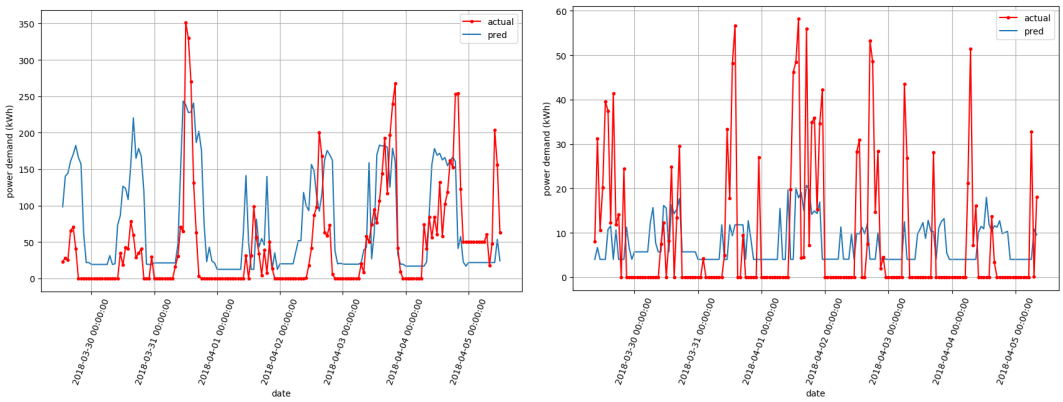
Models	RNN	LSTM	RT	KNN	SVR	AR
Time step 24(1)	0.1179	0.1122	0.1355	0.1036	0.1148	0.1526
Time step 48(2)	0.1598	0.1651	0.1857	0.167	0.1804	0.1733
Time step 72(3)	0.1573	0.1607	0.1834	0.1572	0.1734	0.1383
Time step 96(4)	0.1517	0.1543	0.1652	0.1548	0.1626	0.1527
Time step 120(5)	0.1585	0.1602	0.2048	0.1777	0.1649	0.1299
Time step 144(6)	0.1597	0.1625	0.1882	0.1761	0.1633	0.1441
Time step 168(7)	0.1425	0.1434	0.2338	0.1503	0.1389	0.1708
Overall mean	0.1496	0.1512	0.1852	0.1552	0.1569	0.1517
Computational time	24(sec)	1.3(min)	0.2(sec)	0.3(sec)	7(sec)	0.2(sec)

Figure 6.2 RMSE results for AR vs ML

6.1.3 Test results for Route A and Route B

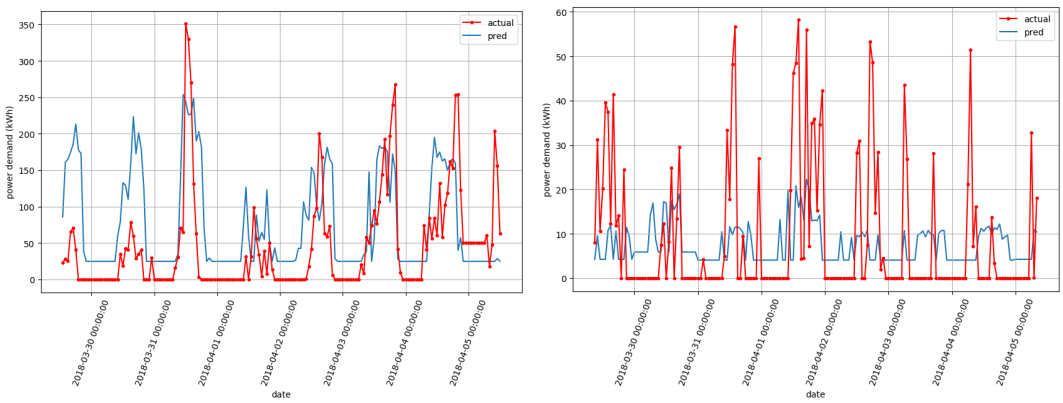
As mentioned, Route A contained the charging stations from File 1 and Route B from File 2 (see Chapter 4). Route A contained nine charging stations, with ID numbers 6, 11, 13, 14, 50, 92, 111, 136, and 160. Route B contained seven charging stations with the ID numbers 35, 37, 55, 105, 130, 162, and 165. Both appear in the Github repository. Charging stations 111 and 136 were removed from Route A in the first test, to focus on the differences between high-activity versus low-activity charging stations. Both tests included the parameters X1–X4 presented in section 5.2.3 (Parameter Selection). The reason for not including parameter X5 in this test was mostly how long it would take to add the X previous hours for both tests. The results for Route A are listed in Table 6.8 and the results for Route B in Table 6.9.

The predictions appear problematic. Comparing the RMSE table and plots showed a difference in results. The RMSE values where very low indicating of very good accuracy, and the plots accuracy showed otherwise. This can be seen particularly in each model's Route B prediction, where the error appears larger than in the RMSE table. One possibility may be the number of zero values. For example, the training data might have many zero values, and thus giving an indication to the models that predicting values close to zero is smart because there might be a percentage chance for it to be zero; this scenario would suggest a better accuracy than what occurs in reality. Figures 6.4, 6.5, 6.6 and 6.7 show each model's ability to predict the two routes.



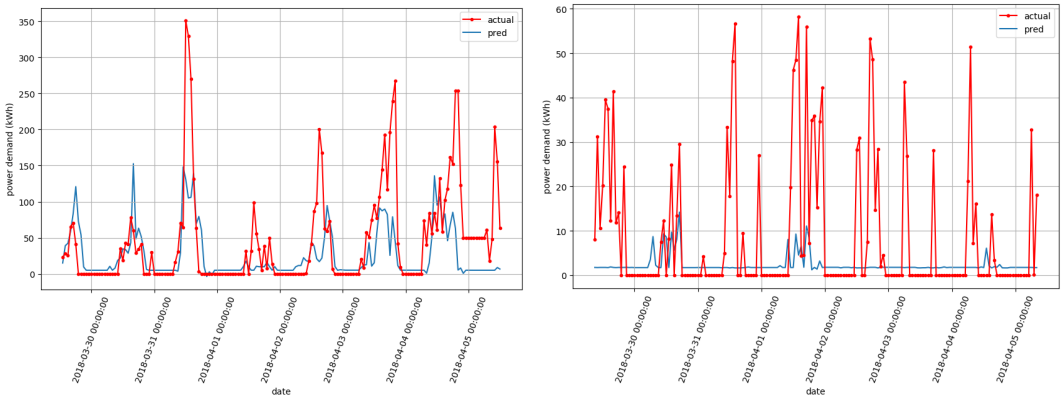
(a) Route A (b) Route B

Figure 6.3 RNN



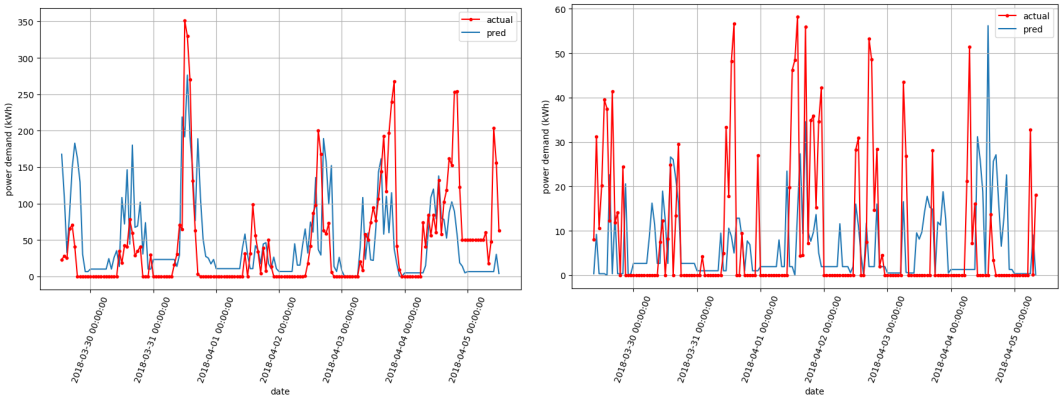
(a) Route A (b) Route B

Figure 6.4 LSTM



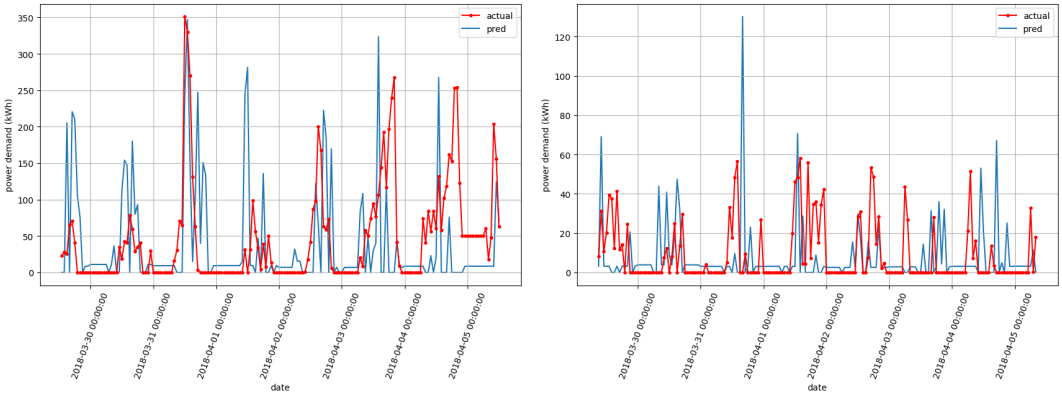
(a) Route A (b) Route B

Figure 6.5 SVR



(a) Route A (b) Route B

Figure 6.6 KNN



(a) Route A (b) Route B

Figure 6.7 RT

Another factor that might have caused the models to err when predicting the values for Route B is relatively low activity. If power consumption is large, it would be relatively easy to differentiate between regular consumption and zero values. That said, the plots are for the 7th (168th) time step and used the validation data to predict, whereas the RMSE table values show the mean of every RMSE value obtained throughout the training, testing and validation. This could explain why the plots and RMSE tables did not correlate. The two models that showed this problem the most were LSTM and RNN, including all other Route B plot predictions. SVR, LSTM and RNN did not understand the pattern clearly, as evident in the amount of time used; this might indicate the data were hard to understand. Hence the models seemed to plot the prediction for charging station 160 better than any other charging stations in both routes, which will be inspected further in the subsection called "After-test summary" (6.1.5).

We inspected the DL models by decreasing the size of the models. We speculated that large networks are complex and finding a pattern was too much work; thus, it might be better to use a smaller network. This idea turned out to be incorrect as their ability was the same no matter the size of the network. Another factor that might have impacted the DL models were how they read the data. Reading all the data – or for a long period – at once might be the wrong way to feed the models data. A better way would be to feed the models only the three previous months; this could be why KNN performed well. KNN looks at the 150 closest neighbors, which has proven to be a big factor in solving time series problems. Another indication that SVR, RNN and LSTM struggled to understand the pattern of these routes was the computational time used. They all required substantial time to make highly inaccurate predictions. KNN had the best overall accuracy for all RMSE values for the prediction of Route A. The results for Route B were strongly affected by the error that might have been caused by zeroes and low-power consumption.

Models	**RNN**	**LSTM**	**RT**	**KNN**	**SVR**
Time step 24(1)	0.1231	0.1051	0.1037	0.0911	0.0929
Time step 48(2)	0.1125	0.113	0.1036	0.0914	0.093
Time step 72(3)	0.1087	0.1037	0.1038	0.092	0.093
Time step 96(4)	0.1114	0.1053	0.1042	0.0923	0.0932
Time step 120(5)	0.1099	0.1104	0.1048	0.0927	0.0931
Time step 144(6)	0.1107	0.1092	0.1049	0.0931	0.0935
Time step 168(7)	0.1476	0.1492	0.2026	0.1326	0.1303
Overall mean	0.1177	0.1137	0.1182	0.0979	0.0984
Computational time	28(min)	108(min)	4(sec)	5(min)	236(min)

Figure 6.8 RMSE results for Route A

Models	RNN	LSTM	RT	KNN	SVR
Time step 24(1)	0.0858	0.0845	0.1034	0.0898	0.0874
Time step 48(2)	0.0852	0.0846	0.1034	0.0902	0.0879
Time step 72(3)	0.0854	0.0854	0.1035	0.0914	0.0881
Time step 96(4)	0.0862	0.0853	0.1036	0.0913	0.0882
Time step 120(5)	0.0856	0.0851	0.1024	0.0923	0.0882
Time step 144(6)	0.0862	0.0849	0.1041	0.0942	0.0883
Time step 168(7)	0.089	0.0887	0.1234	0.1001	0.0971
Overall mean RMSE	0.0862	0.0855	0.1062	0.0927	0.0893
Computational time	38(min)	65(min)	4(sec)	3(min)	58(min)

Figure 6.9 RMSE results for Route B

6.1.4 Parameter test results

As discovered from the above test, most models were relatively effective at predicting the power-demand for charging stations that had a large popularity or were over a certain threshold; none of these occurred in Route B. Therefore, charging station 160 from Route A was used throughout the tests below. The tests started with a prediction made using only the main parameter X1 (max kWh) to create a standard for how well each model performed, without any of the other parameters. Table 6.10 was the standard for us to identify the effects of each parameter, as the table shows RMSE using only parameter X1.

Models	RNN	LSTM	RT	KNN	SVR
Time step 24(1)	0.1253	0.1252	0.171	0.135	0.1268
Time step 48(2)	0.1333	0.1326	0.1781	0.1436	0.135
Time step 72(3)	0.1324	0.1325	0.178	0.1424	0.1345
Time step 96(4)	0.1344	0.1348	0.1805	0.1434	0.1359
Time step 120(5)	0.1342	0.1343	0.1794	0.1455	0.1361
Time step 144(6)	0.1268	0.1275	0.1695	0.1365	0.1289
Time step 168(7)	0.1424	0.1424	0.2348	0.1474	0.137
Overall mean RMSE	0.1327	0.1328	0.1845	0.142	0.1335
Computational time	2(min)	4(min)	0.2(sec)	1(sec)	17(sec)

Figure 6.10 RMSE results for X1 alone

X2 Week day binary

The week-day binary was included to help the models identify the different days. Table 6.11 indicates that this parameter benefited all the models. The decrease in RMSE was not large, but because all models benefited, it was a useful parameter to include to increase the accuracy of the models tested here.

There are no indications for improvements in predicting certain time steps. If certain time steps alone were to benefit from this parameter, then that would indicate that the models were able to identify the pattern for a certain day better than for other days. Since there was an overall decrease in RMSE it indicates an overall effect, with no specific day being easier to predict. This result may also be relevant to this charging station alone, but k-fold cross validation was also used and so the likelihood of that is slim.

Models	RNN	LSTM	RT	KNN	SVR
Time step 24(1)	0.1237	0.1227	0.1642	0.1269	0.1224
Time step 48(2)	0.1292	0.1286	0.1706	0.1342	0.1294
Time step 72(3)	0.1302	0.1315	0.1714	0.1338	0.1311
Time step 96(4)	0.1339	0.1349	0.1758	0.1389	0.1339
Time step 120(5)	0.1327	0.1335	0.1765	0.138	0.1334
Time step 144(6)	0.1262	0.1262	0.1649	0.1302	0.1247
Time step 168(7)	0.1457	0.1505	0.2385	0.1427	0.1355
Overall mean RMSE	0.1316	0.1326	0.1803	0.1349	0.1301
Computational time	2(min)	6(min)	2(sec)	4(sec)	22(sec)

Figure 6.11 RMSE results for week day binary

X3 Peak day binary

After analyzing the data, we added a binary for days that usually displayed high activity, known as peak days. Table 6.12 shows no significant effects for RNN, LSTM, and RT, but slightly benefited KNN and SVR at all time steps and thus also benefited the overall RMSE. There was no clear reason why this parameter had a minimal effect, since prior analysis of the data had indicated that peak days do occur and that they display a pattern. The reason behind the insignificant effect of this parameter might be the lack of precision in adding the binary for the correct peak days. The approach of adding a binary only for certain days that surpass a daily threshold might have benefited the models more.

Models	RNN	LSTM	RT	KNN	SVR
Time step 24(1)	0.1234	0.1234	0.1697	0.1334	0.1251
Time step 48(2)	0.1307	0.1299	0.1751	0.139	0.1306
Time step 72(3)	0.1334	0.1343	0.1773	0.1413	0.1344
Time step 96(4)	0.1349	0.1366	0.1819	0.1428	0.1358
Time step 120(5)	0.1354	0.1366	0.1812	0.1438	0.1362
Time step 144(6)	0.129	0.1298	0.1698	0.1355	0.1288
Time step 168(7)	0.1454	0.1476	0.2354	0.1453	0.1357
Overall mean RMSE	0.1332	0.134	0.1843	0.1402	0.1324
Computational time	3(min)	7(min)	0.2(sec)	2(sec)	18(sec)

Figure 6.12 RMSE results for peak day binary

X4 Temperature

The temperature data were a dummy variable; this aspect was an experiment to see whether including such data would improve the prediction. The data were divided into two parts: 10 degrees for April–September and -10 for October–March.

Table 6.13 shows that KNN benefited the most from adding temperature, but SVR and RT also displayed an overall decrease in RMSE. RNN declined in accuracy from using this parameter and displayed a significant increase in RMSE, whereas LSTM showed a slight increase in RMSE. Overall, temperature was not beneficial to use with the DL models but could slightly improve accuracy when using KNN, SVR and RT.

This testing might be inconclusive and might not actually prove anything in specific. Using a static value for half the year is not realistic and could confuse the models, thus becoming noise to cancel out. Using actual weather data for the days would be far more informative and ideal for testing whether it would improve the accuracy of the models.

Models	RNN	LSTM	RT	KNN	SVR
Time step 24(1)	0.2136	0.1303	0.1643	0.1261	0.1218
Time step 48(2)	0.1897	0.1328	0.1685	0.1334	0.1296
Time step 72(3)	0.1908	0.1329	0.1697	0.1323	0.13
Time step 96(4)	0.1549	0.1352	0.1701	0.1335	0.1317
Time step 120(5)	0.1506	0.1342	0.1721	0.1358	0.132
Time step 144(6)	0.1454	0.1297	0.1612	0.1291	0.1252
Time step 168(7)	0.1414	0.1449	0.2454	0.1532	0.1424
Overall mean RMSE	0.1695	0.1343	0.1788	0.1348	0.1304
Computational time	3(min)	7(min)	0.2(sec)	2(sec)	18(sec)

Figure 6.13 RMSE results for temperature

X5 kWh of X previous hours

To deduce the effects of varying (X) amount of hours on the prediction of the models, this test included X = 24 previous hours as well as X = 168 hours. The results provided insight on how including the previous day or week affected the prediction. Table 6.14 shows the results for using the parameters X1 and the 24 previous hours. Compared to 6.10, Table 6.14 indicates that using the 24 previous hours in the prediction had differing effects on the models. The RNN and LSTM models, that is the DL models, did not benefit from this parameter and showed an overall slight increase in RMSE. RT was somewhat the same, with a small decrease in RMSE. The KNN and SVR models both has an overall decrease in RMSE of 0.02, which indicates a positive effect for these two models.

The results of including the previous week's kWh in the prediction are displayed in Table 6.15. All the models except LSTM were affected positively and displayed a decrease in RMSE. LSTM also showed a slight decrease in RMSE for time steps 2–5 but an increase for the 1st, 2nd and 7th time steps. The RNN model displayed a slight shift towards being positively affected by including this parameter. RT, KNN and SVR had substantial decreases in RMSE, indicating that including this parameter increased the accuracy of these models.

This test showed that the SVR and KNN models benefitted the most among all the models. This result evoked a question about whether including even older hours might help the accuracy. Under further inspection, including 336 previous hours (i.e. the last two weeks) led to a slight increase in RMSE compared to including the 168 previous hours. These results indicated that including only the previous week was sufficient and using further data might decrease the benefits of including previous observations. The results show that including the previous week (i.e. the 168 previous observations of max kWh) increased the accuracy of all models but especially RT, KNN and SVR. For RNN and LSTM, the increase in accuracy was minimal but is still a benefit if these models are used in further research.

Models	RNN	LSTM	RT	KNN	SVR
Time step 24(1)	0.1336	0.132	0.1689	0.1192	0.1172
Time step 48(2)	0.1405	0.1374	0.1738	0.124	0.1231
Time step 72(3)	0.1388	0.136	0.1771	0.1259	0.1221
Time step 96(4)	0.1414	0.1411	0.1823	0.129	0.1248
Time step 120(5)	0.1373	0.1383	0.1764	0.1242	0.1212
Time step 144(6)	0.1154	0.1145	0.1603	0.1124	0.1093
Time step 168(7)	0.1528	0.1588	0.2398	0.1364	0.1373
Overall mean RMSE	0.1371	0.1369	0.1827	0.1244	0.1222
Computational time	2(min)	5(min)	19(sec)	22(sec)	18(sec)

Figure 6.14 RMSE results for the 24 previous hours

Models	RNN	LSTM	RT	KNN	SVR
Time step 24(1)	0.1232	0.1353	0.145	0.1088	0.1022
Time step 48(2)	0.1244	0.1255	0.1505	0.1099	0.1033
Time step 72(3)	0.1272	0.1282	0.1558	0.1099	0.1031
Time step 96(4)	0.1301	0.1232	0.1553	0.1108	0.104
Time step 120(5)	0.1272	0.1222	0.1536	0.1112	0.1042
Time step 144(6)	0.129	0.1299	0.1534	0.1114	0.1035
Time step 168(7)	0.1457	0.1976	0.2418	0.138	0.1368
Overall mean RMSE	0.1296	0.1374	0.165	0.1143	0.1082
Computational time	3(min)	6(min)	2(min)	3(min)	4(min)

Figure 6.15 RMSE results for the 168 previous hours

6.1.5 After-test summary

A problem was brought to our attention when we conducted the tests. The plots and the RMSE values did not correlate as previously stated. The RMSE values too low for plots showing at places an accuracy of roughly 40%. We suspected the zero values included in the data for being the cause of this. Charging station 160 showed better plots than the other charging stations and we wanted to learn why this was the case.

Thus, knowing how many zeroes occurred for each day in each route would help us to verify whether the zero values were a problem. Figures 6.16 and 6.17 show the number of zeros for each route. Both routes had on average equal zero values; we could not know exactly why the plots were better for charging station 160. On further investigation, charging station 160 had no zero values at all; no other charging station had no zero values. The only stations that came close were charging station 162 in File 2 and 92 in File 1, which both had four zero values in their data.

We found it strange that these three charging stations had so few zero values. The analysis plots for each charging station were recreated with the code found in "file dataAnalysis.py" under the function name "dayPattern". After performing further predictions for charging stations 160, 92 and 162, we noted that the number of zero values was not strongly related to the accuracy of the plot. The RMSE value was related and low power consumption aided the RMSE values. This led us to assume that the higher the activity of the charging station, the easier it was to plot. This was based on the fact that the only difference between the charging stations mentioned was that charging station 160 had higher activity than the others.

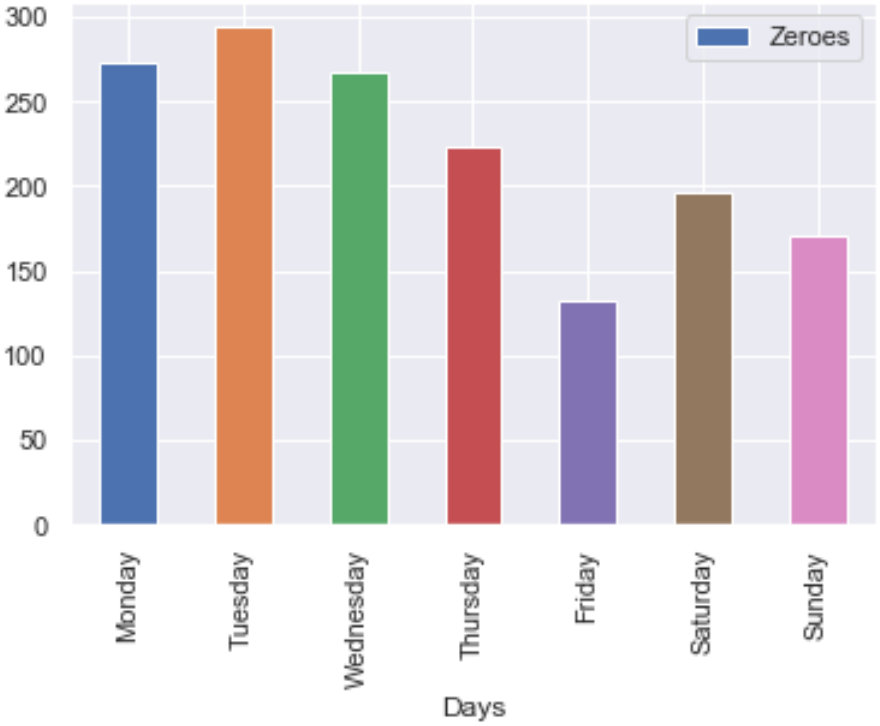


Figure 6.16 Number of zeroes per day in file 1

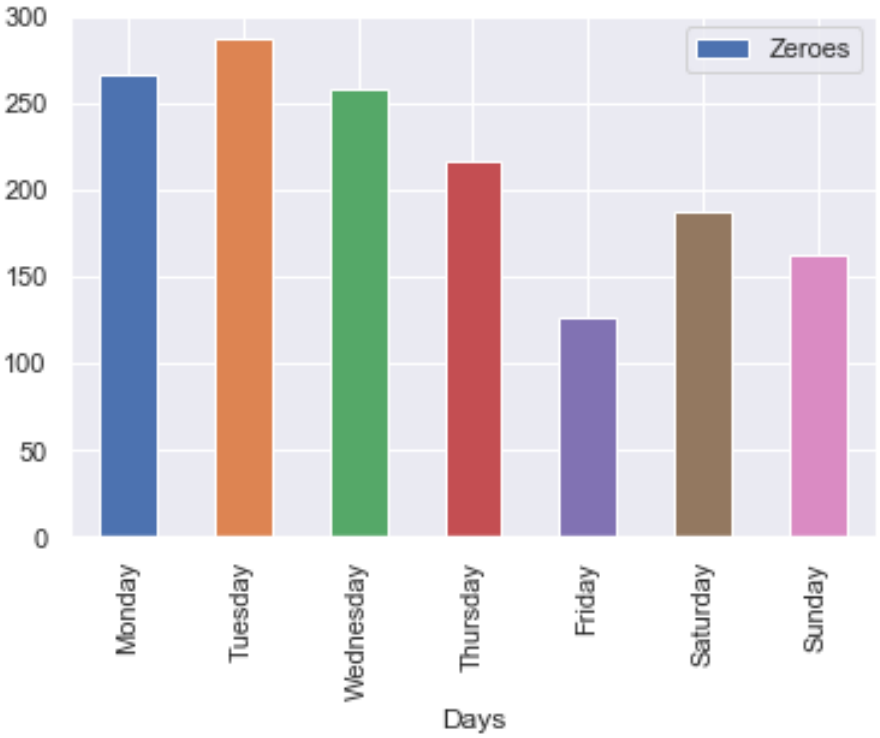


Figure 6.17 Number of zeroes per day in file 2

Chapter 7

Conclusion and further work

This chapter offers some concluding remarks regarding the results. It also suggests recommendation for future research and real-time learning.

7.1 Conclusion

In this thesis, three major tests were conducted on different algorithms. The algorithms were compared for their ability to predict the power flows of EV charging stations at several important locations in Norway.

In the first test, six algorithms were tested to compare classical methods such as AR and ML models. The test was conducted with univariate input data (only kWh) and the data was for a high-activity EV charging station (ID 160). The six algorithms were AR, RNN, LSTM, SVR, RT and KNN. RNN achieved the best overall accuracy score for RMSE, with LSTM and AR in close second place. It should be noted that this test can lack confidence, mainly due to the use of only one evaluation metric. Combining walk-forward validation with RMSE in this test would have increased the confidence.

The second test had the objective of comparing the abilities of the models RNN, LSTM, SVR, RT and KNN to predict two different routes in Norway. The routes had different activity levels; Route A is a much more popular route and thus incurs higher power consumption than the less popular Route B. A problem that emerged from this test was that the RMSE values were more accurate than the results which were plotted in the graphs. We speculated that this was the caused by the large number of zero values in the data set. This meant that every test was affected by this and that it showed better accuracy than the models actually achieved. Nevertheless, we could still determine which models benefited the most. Based on the results from Route A, which seemed to be less affected by zero values, KNN achieved the greatest accuracy. Route B predictions were strongly affected and most of the plots showed this pattern. However, there might be more than one factor to explain why the predictions were affected as they were. There were indications that both the

number of zero values in the data and the power-consumption levels could have influenced the results. The combination of these two factors might have been why Route B plot predictions proved to be inaccurate.

The third test had the objective of discovering how each parameter mentioned in Chapter 5 (Parameter Selection) affected the models RNN, LSTM, SVR, RT and KNN. The previous week's observations (i.e. 168 previous observations) was the most useful parameter. Weekday binary also had a positive effect on the accuracy of the models. The temperature parameter showed beneficial effects for KNN, RT and SVR, but not for LSTM and RNN; the latter two models showed worse RMSE scores when this parameter was included (especially RNN). Peak-day binary led to less accurate scores among all the parameters. This was probably because of how the parameter were implemented into the data, thus some adjusting to how the peak-day binary is added to the data is advised.

Further analysis of the number of zero values indicated that these might not be the reason for the inaccuracy of the plots but rather why the RMSE values were accurate. Another finding was that high-activity consumption might be easier to predict than low power consumption.

7.2 Future work

Although this thesis did not identify a superior model, there are some indications for future avenues of research. The next sections contain future recommendations for treating of the data, multi-step strategy, model choice and parameter choice. The last subsection contains a real-time learning relevant literature review.

7.2.1 Initial data analysis

When we preprocessed the data and wanted to include a binary variable for peak days, we had to re-analyse the data to find the peak days. Returning for further analysis happened on multiple occasions in the preprocessing of the data and development of the models. We strongly advice to perform the analysis of the data in intervals with future implementations.

7.2.2 Multi-step strategy

We experimented with all of the multi-step strategies presented in the second chapter, but found a customized version would fit most of the models without interfering with how they worked. Thus, we customised a strategy to suit all

models without changing them; later, this became a recursive strategy without the previous predictions.

The four strategies described in Chapter 2 were used and tested and did not yield any major differences in the results. Any one of them could have been used, depending on the research purpose. Using the direct strategy would remove all bias in the models when predicting the next time step; this approach is also easily implemented. Depending on the model choice, a RECTIFY hybrid could be beneficial, especially for models capable of handling many parameters at once. It removes the bias but includes previous predictions.

7.2.3 Data batching

Structuring the data into batch sizes might help in predicting a certain period at a time. Batching the data into days or weeks helps the models to predict in days or weeks, which would require fewer time steps. That is, rather than predicting the 168th time step in the future, as done in this thesis, one would predict only seven time steps or one time step into the future because the data were batched that way.

7.2.4 Model recommendation

As noted throughout most of the tests, KNN proved to have the best accuracy in most cases. This was evident in that it benefited from most of the parameters while requiring a low computational time. That said, KNN is not a superior model and the choice of model would depend on the direction, objectives, size and complexity of the problems being addressed.

7.2.5 Parameter recommendations

We recommend using the parameters: previous week and weekday binary as is for further research. Peak-day binary needs some re-adjustment as previously mentioned. Some models benefited from the temperature dummy data, but it would be advised to test with real data for a more confident deduction. Lastly, a study, [48], analysing and reviewing models predicting commercial building electricity load seem to have been using previous observations differently in their prediction. The study included “previous day same hour” and “previous week same hour loads" and “previous 24-h average load” as parameters. Including these parameters might be more relevant than the peak-day parameter.

7.2.6 Real-time learning

The trends regarding EVs and charging infrastructure indicate that data are becoming increasingly available regarding both EVs and charging stations. The transfer of data will most likely happen close to real time. Therefore, if models can benefit from the newly received data instantaneously, highly reliable and customised predictions become attainable.

Using ML models in an online setting is a new aspect of the ML domain and encounters many difficulties, especially considering real-time learning. There are issues with using ML models in an online setting, such as training the model with new data. To react to new data and learn over time, ML practitioners typically schedule training the model on newer data and automatically deploy it once a week, for instance. The problem here is that even if one trains the model each week, the model will lag because it is trained on stale data. The ideal case is to train the model using new data in real time, thus not only predicting in real time but also learning in real time.

Yet again, there are unaddressed concerns in using ML models in real time. Most of the solutions for overfitting, underfitting and so on that are used in a closed environment (not an online setting) are not applicable in a real-time setting. Therefore, establishing the appropriate requirements for a real-time learning setting is important. Machine learning for real-time learning and prediction of power demand for charging stations is a new field in which little research has been conducted. In addition to this specific lack, little research has been conducted on ML for real-time prediction and learning in general.

A study [33] adopted an ML approach to predict future power demand in real-time for a battery-operated car. While performing the prediction, the researchers discovered that the battery depended on far more than they had expected; therefore, they had to construct a self-corrective regression model. Another study [27] was aimed at predicting severe complications during critical care in real time after cardiothoracic surgery. The researchers used DL models, specifically RNN, for real-time prediction. Their study resulted in the DL models outperforming the standard clinical reference tools.

Achieving real-time ML and DL prediction might face an extreme processing bottleneck because the data are typically stored. Performing analysis on these data using traditional ML and DL methods usually requires extracting and then transforming and loading the data. Terry Erisman in the article [8], proposes a solution to this challenge with an open source in-memory computing platform called Apache Ignite 2.4.

Regarding access to data from both EVs and charging stations, the capacity to handle many parameters as input data (i.e. handling multivariate input data) seems to be beneficial in terms of data availability. This requirement renders sophisticated and complex algorithms, such as ML and DL, more powerful and effective in this

aspect than AR models. However, AR models might prove to have better accuracy when predictions are based on univariate input data.

Bibliography

[1] Ayodele Ariyo Adebiyi, Aderemi Oluyinka Adewumi, and Charles Korede Ayo. "Comparison of ARIMA and artificial neural networks models for stock price prediction". In: *Journal of Applied Mathematics* 2014 (2014).

[2] Nima Amjady. "Short-term hourly load forecasting using time-series modeling with peak load estimation capability". In: *IEEE Transactions on Power Systems* 16.3 (2001), pp. 498–505.

[3] *Autoregressive models*. https://deepgenerativemodels.github.io/notes/autoregressive/. Last Accessed: 07-07-2019.

[4] Jatin Bedi and Durga Toshniwal. "Empirical mode decomposition based deep learning for electricity demand forecasting". In: *IEEE Access* 6 (2018), pp. 49144–49156.

[5] George EP Box et al. *Time series analysis: forecasting and control*. John Wiley & Sons, 2015.

[6] Bo-Juen Chen, Ming-Wei Chang, et al. "Load forecasting using support vector machines: A study on EUNITE competition 2001". In: *IEEE transactions on power systems* 19.4 (2004), pp. 1821–1830.

[7] Vincent Debusschere, Seddik Bacha, et al. "One week hourly electricity load forecasting using neuro-fuzzy and seasonal ARIMA models". In: *IFAC Proceedings Volumes* 45.21 (2012), pp. 97–102.

[8] *Achieving real-time machine learning and deep learning with in-memory computing*. https://jaxenter.com/in-memory-computing-machine-learning-145623.html. Last Accessed: 10-10-2019.

[9] Douglas Eck and Juergen Schmidhuber. "Finding temporal structure in music: Blues improvisation with LSTM recurrent networks". In: *Proceedings of the 12th IEEE workshop on neural networks for signal processing*. IEEE. 2002, pp. 747–756.

[10] *page 2 EV Market Shares*. https://www.toi.no/getfile.php?mmfileid=40780. Accessed: 2019-01-01.

[11] Cheng Fan, Fu Xiao, and Shengwei Wang. "Development of prediction models for next-day building energy consumption and peak power demand using data mining techniques". In: *Applied Energy* 127 (2014), pp. 1–10.

[12] *GHG Emissions Sources of Greenhouse Gas Emissions*. https://www.epa.gov/ghgemissions/sources-greenhouse-gas-emissions. Accessed: 2019-01-01.

[13] Alex Graves, Abdel-rahman Mohamed, and Geoffrey Hinton. "Speech recognition with deep recurrent neural networks". In: *2013 IEEE international conference on acoustics, speech and signal processing*. IEEE. 2013, pp. 6645–6649.

[14] Geoffrey E Hinton, Simon Osindero, and Yee-Whye Teh. "A fast learning algorithm for deep belief nets". In: *Neural computation* 18.7 (2006), pp. 1527–1554.

[15] Sepp Hochreiter and JĂźrgen Schmidhuber. "Long short-term memory". In: *Neural computation* 9.8 (1997), pp. 1735–1780.

[16] David Howell et al. "Enabling Fast Charging: A Technology Gap Assessment". In: *technical peer-reviewed journals* (Oct. 2017). DOI: 10.2172/1416167.

[17] JosĂŠ M Jerez et al. "Missing data imputation using statistical and machine learning methods in a real breast cancer problem". In: *Artificial intelligence in medicine* 50.2 (2010), pp. 105–115.

[18] D Koschwitz, J Frisch, and C van Treeck. "Data-driven heating and cooling load predictions for non-residential buildings based on support vector machine regression and NARX Recurrent Neural Network: A comparative study on district scale". In: *Energy* 165 (2018), pp. 134–142.

[19] Dimitris Lazos, Alistair B Sproul, and Merlinde Kay. "Optimisation of energy management in commercial buildings with weather forecasting inputs: A review". In: *Renewable and Sustainable Energy Reviews* 39 (2014), pp. 587–603.

[20] Chen Lin and Min Chi. "A comparisons of bkt, rnn and lstm for learning gain prediction". In: *International Conference on Artificial Intelligence in Education*. Springer. 2017, pp. 536–539.

[21] *Multi-step Time Series Forecasting with Long Short-Term Memory Networks in Python*. https://machinelearningmastery.com/multi-step-time-series-forecasting-long-short-term-memory-networks-python/. Last Accessed: 07-07-2019.

[22] Mostafa Majidpour et al. "A novel forecasting algorithm for electric vehicle charging stations". In: *2014 International Conference on Connected Vehicles and Expo (ICCVE)*. IEEE. 2014, pp. 1035–1040.

[23] Mostafa Majidpour et al. "Fast prediction for sparse time series: Demand forecast of EV charging stations for cell phone applications". In: *IEEE Transactions on Industrial Informatics* 11.1 (2015), pp. 242–250.

[24] Spyros Makridakis, Evangelos Spiliotis, and Vassilios Assimakopoulos. "Statistical and Machine Learning forecasting methods: Concerns and ways forward". In: *PloS one* 13.3 (2018), e0194889.

[25] *Applied predictive modeling*. https://vuquangnguyen2016.files.wordpress.com/2018/03/applied-predictive-modeling-max-kuhn-kjell-johnson_1518.pdf. Last Accessed: 06-09-2019.

[26] Andrew V Metcalfe and Paul SP Cowpertwait. *Introductory time series with R*. Springer, 2009, pp. 2, 6.

[27] Alexander Meyer et al. "Machine learning for real-time prediction of complications in critical care: a retrospective study". In: *The Lancet Respiratory Medicine* 6.12 (2018), pp. 905–914.

[28] *Machine learning definitions*. https://emerj.com/ai-glossary-terms/what-is-machine-learning/. Last Accessed: 07-07-2019.

[29] *How to Develop Multi-Step LSTM Time Series Forecasting Models for Power Usage*. https://machinelearningmastery.com/how-to-develop-lstm-models-for-multi-step-time-series-forecasting-of-household-power-consumption/. Last Accessed: 07-07-2019.

[30] Alex D Papalexopoulos and Timothy C Hesterberg. "A regression-based approach to short-term system load forecasting". In: *IEEE Transactions on Power Systems* 5.4 (1990), pp. 1535–1547.

[31] John Patten et al. "The impact of temperature on plug-in hybrid electric vehicle battery performance". In: *2011 IEEE Vehicle Power and Propulsion Conference*. IEEE. 2011, pp. 1–2.

[32] *What does Peak shaving / load shifting mean?* https://www.next-kraftwerke.com/knowledge/what-is-peak-shaving. Last Accessed: 07-07-2019.

[33] Somnath Pradhan and Joydeb Roychaudhury. "A machine learning approach to predict future power demand in real-time for a battery operated car". In: *2014 International Conference on the IMpact of E-Technology on US (IMPETUS)*. IEEE. 2014, pp. 49–56.

[34] Ramu Ramanathan et al. "Short-run forecasts of electricity loads and peaks". In: *International journal of forecasting* 13.2 (1997), pp. 161–174.

[35] *Page 14 Recharging in Cities*. https://www.transportenvironment.org/sites/te/files/Charging%20Infrastructure%20Report_September%202018_FINAL.pdf. Accessed: 2019-01-01.

[36] *page 2 EV world record 45.3%*. https://ofv.no/bilsalget/bilsalget-i-september-2018. Accessed: 2019-01-01.

[37] Johannes Rolink and Christian Rehtanz. "Estimation of the availability of grid-connected electric vehicles by non-homogeneous semi-Markov processes". In: *2011 IEEE Trondheim PowerTech*. IEEE. 2011, pp. 1–7.

[38] Nor Erniza Mohammad Rozali et al. "Peak-off-peak load shifting for hybrid power systems based on Power Pinch Analysis". In: *Energy* 90 (2015), pp. 128–136.

[39] MarÄąĚa Ruiz-AbellĂłn, Antonio GabaldĂłn, and Antonio GuillamĂłn. "Load forecasting for a campus university using ensemble methods based on regression trees". In: *Energies* 11.8 (2018), p. 2038.

[40] Galit Shmueli and Kenneth C Lichtendahl. *Practical time series forecasting: A hands-on guide*. Axelrod Schnall Publishers, 2016.

[41] Robert H Shumway and David S Stoffer. *Time series analysis and its applications: with R examples*. Springer, 2017.

[42] Souhaib Ben Taieb, Rob J Hyndman, et al. *Recursive and direct multi-step forecasting: the best of both worlds*. Vol. 19. Citeseer, 2012.

[43] Ruey S Tsay. *Multivariate time series analysis: with R and financial applications*. John Wiley & Sons, 2013.

[44] Moslem Uddin et al. "A review on peak load shaving strategies". In: *Renewable and Sustainable Energy Reviews* 82 (2018), pp. 3323–3332.

[45] Mark W Verbrugge and Charles W Wampler. "On the optimal sizing of batteries for electric vehicles and the influence of fast charge". In: *Journal of Power Sources* 384 (2018), pp. 312–317.

[46] Paul J Werbos et al. "Backpropagation through time: what it does and how to do it". In: *Proceedings of the IEEE* 78.10 (1990), pp. 1550–1560.

[47] Ronald J Williams and David Zipser. "A learning algorithm for continually running fully recurrent neural networks". In: *Neural computation* 1.2 (1989), pp. 270–280.

[48] Baran Yildiz, Jose I Bilbao, and Alistair B Sproul. "A review and analysis of regression and machine learning models on commercial building electricity load forecasting". In: *Renewable and Sustainable Energy Reviews* 73 (2017), pp. 1104–1122.